VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

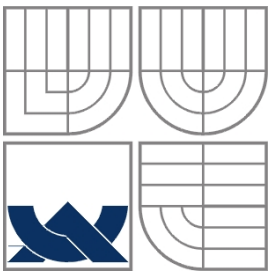
EVOLUCE EMERGENTNÍHO CHOVÁNÍ V CELULÁRNÍCH SYSTÉMECH

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

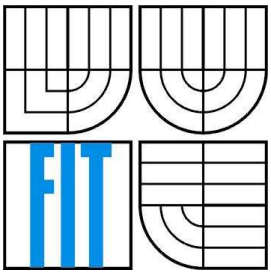
AUTOR PRÁCE
AUTHOR

Bc. RADIM NOVÁK

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

EVOLUCE EMERGENTNÍHO CHOVÁNÍ V CELULÁRNÍCH SYSTÉMECH

EVOLUTION OF EMERGENT BEHAVIOR IN CELLULAR SYSTEMS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. RADIM NOVÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL BIDLO, Ph.D.

BRNO 2010

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2009/2010

Zadání diplomové práce

Řešitel: **Novák Radim, Bc.**
Obor: Počítačové systémy a sítě
Téma: **Evoluce emergentního chování v celulárních systémech**
Evolution of Emergent Behavior in Cellular Systems
Kategorie: Počítačová architektura

Pokyny:

1. Seznamte se s problematikou celulárních automatů (CA).
2. Navrhněte a implementujte simulátor celulárních automatů, který bude snadno použitelný ve vašich experimentech.
3. Zvolte vhodnou oblast (případně oblasti) k demonstraci tzv. emergentního chování v CA (např. replikace struktury, samoorganizace, aplikace výpočtu CA jako generátoru struktur apod.).
4. Nalezněte taková pravidla CA, která povedou k emergentnímu chování. Využijte například evoluční algoritmy.
5. Proveďte experimenty s různými CA, srovnejte výpočetní náročnost s ohledem na zvolenou aplikaci.
6. Diskutujte možnosti využití navržených celulárních automatů.
7. Zhodnoťte dosažené výsledky.

Literatura:

- Podle pokynů vedoucího.

Při obhajobě semestrální části diplomového projektu je požadováno:

- Body 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci ročníkového a semestrálního projektu (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Bidlo Michal, Ing., Ph.D.**, UPSY FIT VUT
Datum zadání: 21. září 2009
Datum odevzdání: 26. května 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Tato práce se zabývá problematikou celulárních automatů a jejich využití v oblasti výzkumu sebe-replikace. Konkrétně se zaměřuje na sebe-replikující se smyčky a ukazuje možné přístupy k optimalizaci jejich schopnosti replikace. První kapitoly jsou teoretickým úvodem do oblasti celulárních automatů, seznamují čitatele s problematikou sebe-replikace v celulárních automatech a představují vybrané sebe-replikující se smyčky, počínaje nejznámější Langtonovou smyčkou. Další části prezentují výsledky práce v oblasti optimalizace sebe-replikace u dvou variant smyček - Bylovy a Chou-Reggia smyčky. Jsou představeny dva přístupy k optimalizaci a jejich možná kombinace. Jeden spočívá v obohacení smyčky o schopnost vícenásobné sebe-replikace. U druhého je cílem optimalizace redukce potřebného počtu kroků k vytvoření repliky smyčky.

Abstract

This master's thesis deals with the topic of cellular automata and their utilization in the research of self-replication, especially with the focus on self-replicating loops. It also shows several possible approaches how to optimize the replication process. The first part is focused on theoretical aspects of cellular automata. It acquaints the readers with the questions of self-replication in the cellular automata and present some of the existing self-replicating loops, starting with the widely known Langton's loop. The second part presents the optimization of the replication process considering two selected variants of self-repricating loops - Byl's loop and Chou-Reggia loop. Two approaches are introduced together with their possible combination. The first approach is based on multiple self-replication. The second one is based on the reduction of the number of steps of the cellular automaton needed to create a copy of the loop.

Klíčová slova

Celulární automat, sebe-replikace, sebe-replikující se smyčky, Langtonova sebe-reprodukující se smyčka, Bylova smyčka, Chou-Reggia smyčka, urychlená sebe-replikace, vícenásobná sebe-replikace.

Keywords

Cellular automata, self-replication, self replicating loops, Langton's self-reproduction loop, Byl's loop, Chou-Reggia loop, accelerated self-replication, multiple self-replication.

Citace

Novák Radim: Evoluce emergentního chování v celulárních systémech, diplomová práce, Brno, FIT VUT v Brně, 2010.

Evoluce emergentního chování v celulárních systémech

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením
Ing. Michala Bidla, Ph.D.
Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

……
Radim Novák
30. května 2010

Poděkování

Chtěl bych zde velmi poděkovat svému vedoucímu Ing. Michalu Bidlovi, Ph.D. za velkou ochotu, pomoc a rady při psaní této práce.

Obsah

1 Úvod

Celulární systémy nacházejí v dnešní době velké aplikační uplatnění, jak v oblasti výzkumu, tak v oblasti praktického nasazení. Asi nejznámějším zástupcem této kategorie je jistojistě celulární automat. Tento v zásadě jednoduchý systém dokáže vykonávat i velice složité výpočty a může být použit k simulaci komplexního chování. Tohoto je dosaženo za použití desítek, stovek až statisíců malých buněk tvořících tento automat, které samostatně neprojevují žádné nebo minimální inteligentní chování, ale v interakci s okolními buňkami může dojít doslova k „vynoření“ i velice komplexního chování. Tento „jev“ označujeme pojmem *emergence*, nebo jako *emergentní chování* (Poznámka: Emergence je pojem značně nejasný a dosud není zavedena žádná univerzální definice. O vysvětlení tohoto pojmu se pokouší esej [35] napsaná G. E. Marshem.).

Jako zástupce emergentního chování jsem si zvolil v celulárních automatech nejdéle řešenou problematiku, a to je sebe-replikace struktur. Sebe-replikace v celulárním automatu je proces, při kterém struktura projevuje takové chování, jehož důsledkem je vytvoření přesné kopie sama sebe. Tato kopie pak dále projevuje stejné chování a provádí další replikaci sama sebe.

Cílem této práce je představit dosavadní vývoj v oblasti sebe-replikace struktur se zaměřením na struktury ve tvaru smyčky, tzv. sebe-replikující se smyčky, a ukázat možné přístupy k optimalizaci jejich schopnosti replikace. Zvolil jsem dvě smyčky, které jsem urychlil za použití dvou odlišných přístupů. První popisovaný přístup je založen na obohacení smyčky o schopnost vícenásobné sebe-replikace. Druhý přístup spočívá v redukci počtu potřebných vývojových kroků k vytvoření repliky smyčky za pomoci rozšíření stavového prostoru celulárního automatu, případně v kombinaci s přístupem prvním. Tyto přístupy a získané výsledky představuje poslední kapitola práce.

Ve 2. kapitole představuji celulární automaty v jejich základním a nejznámějším provedení. Po stručném historickém úvodu pokračuje kapitola neformální i formální definicí, následované dalšími informacemi důležitými k pochopení funkce celulárního automatu a následné aplikace v oblasti sebe-replikace. Seznamuji zde čitatele také s Wolframovou klasifikací celulárních automatů podle typu chování a se slavnou hrou vytvořenou Johnem Conwayem „Game of Life“, jako známým příkladem aplikace celulárních automatů.

3. kapitola se zabývá samotnou problematikou sebe-replikace v celulárních automatech. Vysvětluje důvody k výzkumu v této oblasti a dělí přístupy k řešení problematiky do třech proudů. Následuje zevrubné představení von Neumannova univerzálního konstruktoru, jako otce všech sebe-replikujících se struktur v celulárních automatech a především dalšího zástupce univerzálních výpočetních strojů s výpočetní silou rovnou Turingovu stroji. Dalším představeným zástupcem univerzálních výpočetních strojů je Coddův automat, který se stal inspirací k první a nejznámější sebe-replikující se smyčce vytvořené roku 1984 Christopherem Langtonem. Po ní následuje představení dalších smyček až do současnosti.

Další kapitola představuje přístup k návrhu pravidel celulárního automatu pomocí evolučních technik. Pro účely práce jsem zvolil klasický genetický algoritmus. Je zde představena jeho implementace a dosažené výsledky evoluce. Dále je zde vysvětleno, proč jsem od jejího použití v další fázi práce odstoupil a použil raději přístupu experimentálního (ručního) návrhu.

Součástí diplomové práce bylo také vytvoření simulátoru celulárních automatů, který byl důležitým prostředkem pro provádění experimentů. Simulátor je představen v příloze této práce.

V rámci semestrálního projektu vznikly první dvě kapitoly této práce představující celulární automaty a sebe-replikující se smyčky. Této studie jsem využil k analýze chování smyček a

následnému návrhu pravidel vedoucích k urychlení sebe-replikace smyček, tak jak je to uvedeno v kapitole 5.

2 Celulární automat

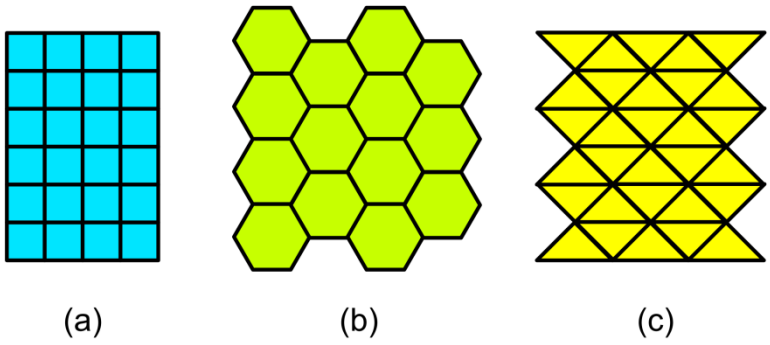
John von Neumann, maďarský matematik židovského původu, se ve 40. letech minulého století začínal zabývat myšlenkou definice života jako logického procesu, blízkému tomu, co by se mohlo odehrávat ve stroji. Jeho cílem bylo vytvořit tak dostatečně komplikovaný stroj, aby se dokázal sám replikovat (rozmnožovat). K tomuto účelu vytvořil *samoreprodukující se automat*. Jedná se o robota, který se pohybuje (pluje) v moři vlastních součástek a dokáže se množit. Matematik Stanislaw Ulam seznámený s von Neumannovou prací navrhnul nahradit plovoucího robota něčím, co dnes nazýváme nejčastěji jako *celulární automat* (nebo také *buněčný automat*, angl. *cellular automata*). [2]

V informatice se můžeme setkat s mnoha synonymy pro tento automat, včetně (anglicky) *tessellation automata*, *cellular spaces*, *iterative automata*, *homogeneous structures* a *universal spaces* [3].

2.1 Neformální popis

Celulární automat (CA) [1] je dynamický systém, tvořený typicky *diskrétní* soustavou buněk, které nabývají hodnot z nějaké (konečné) množiny stavů. Ne tak častou variantou celulárních automatů je *spojitý* CA, kde stavy buněk nenabývají diskrétních hodnot z množiny stavů, ale hodnot použité spojité funkce, např. hodnot z intervalu $\langle 0; 1 \rangle$. Celulární automaty můžou být dále *deterministické* nebo *nedeterministické* (stochastické). Nejčastěji se ale setkáváme s deterministickými a diskrétními celulárními automaty, proto se dále v práci zaměřím pouze na ně.

Jak jsem již psal výše, tak se jedná o dynamický systém, který je diskrétní jak v prostoru, tak v čase. Sestává se z obecně n-rozměrného pole buněk, které můžou nabývat stavů z konečné množiny stavů. Pole buněk mívá typicky mřížovou strukturu, tj. buňky jsou např. čtvercové (nejčastěji) a jsou zarovnány do struktury tvořící obdélník (pro 2 dimenze), nebo kvádr (pro 3 dimenze), atp. Můžeme se ale setkat i s jinými strukturami tvořenými jinými typy buněk, např. hexagonální buňky, nebo trojúhelníkové buňky (viz. následující obrázek).

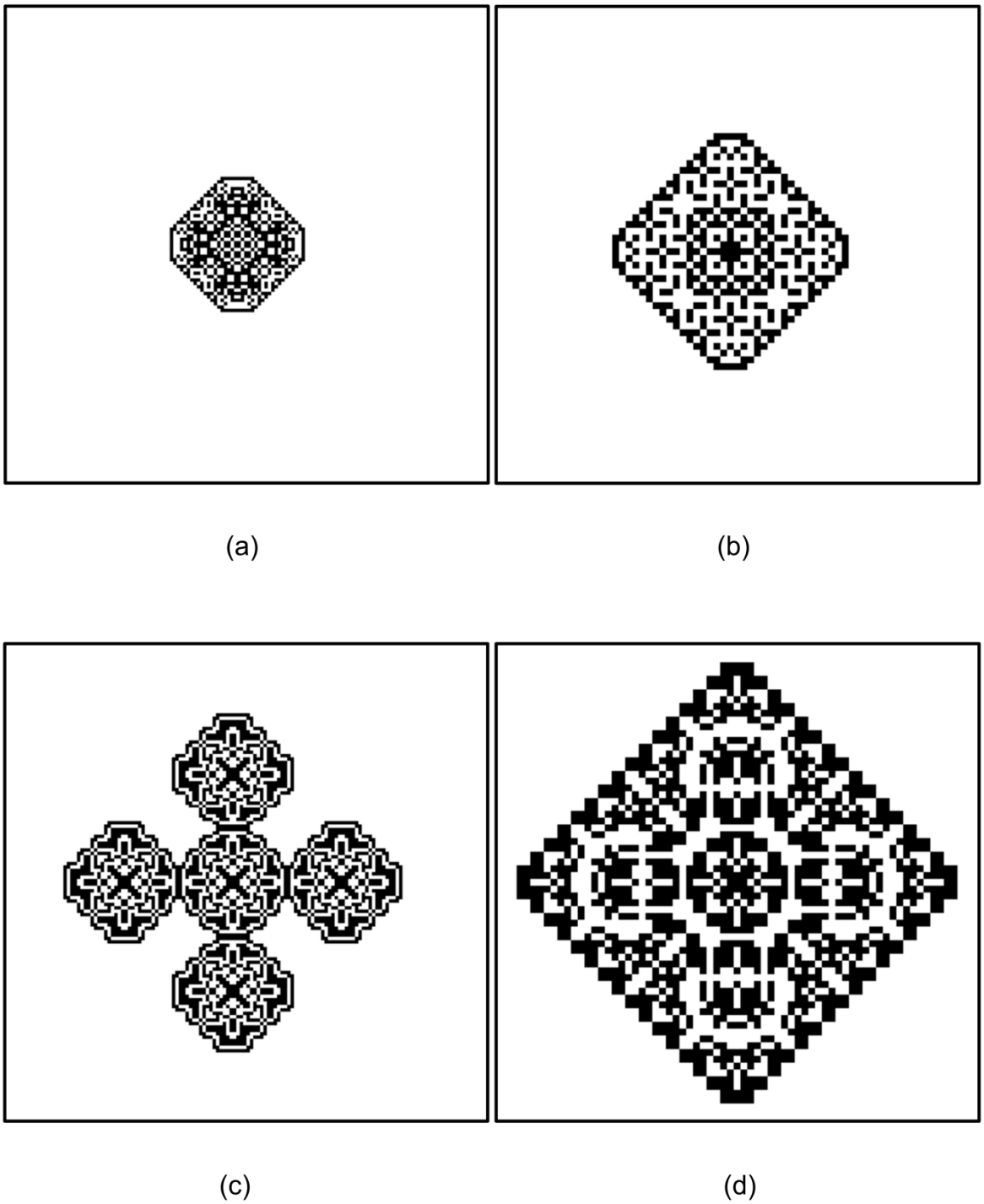


Obrázek 2.1: Příklady struktur celulárního automatu. (a) klasická struktura, čtvercové buňky, (b) hexagonální buňky, (c) trojúhelníkové buňky

Stavy buněk se mění na základě lokálních přechodových pravidel. Pravidla jsou obecně funkce stavu buňky a jejího okolí definující nový stav buňky v čase. Pravidlo může být zadáno různým způsobem. Například pomocí jednoduché logické funkce (AND, OR, XOR, atp.), zápis pomocí

tabulky obsahující pro všechny kombinace okolí nový stav, nebo pomocí soustavy složitých diferenciálních rovnic.

Systém je diskrétní v čase, tzn. že buňky mění svůj stav na základě přechodového pravidla v diskrétních časových krocích. Buňky CA navíc mění svůj stav paralelně – jedná se o *masivně paralelní* systém.



Obrázek 2.2: Ukázka vývinu 2D CA. Bílé čtverečky představují stav 0, černé stav 1. (a) počáteční stav, tj. t = 0, (b) t = 30, (c) t = 60, (d) t = 90 (obrázek převzat z [1])

2.2 Formální definice

Následující definice celulárního automatu vychází z [1]. Jedná se o definici modelu CA podle Edgara F. Codda (1968). Z důvodu zjednodušení problému bude CA definován na 2-dimenzionální variantě.

Nechť I představuje množinu celých čísel. Abychom obdrželi celulární prostor, tak asociujeme s množinou $I \times I$ následující:

1. *Funkci sousedství* $g: I \times I \rightarrow 2^{I \times I}$, definovanou předpisem

$$g(\alpha) = \{\alpha + \delta_1, \alpha + \delta_2, \dots, \alpha + \delta_n\}, \tag{2.1}$$

   pro všechny $\alpha \in I \times I$, kde $\delta_i \in I \times I$ $(i = 1, 2, \dots, n)$. δ_i označují pozici sousedních buněk a mají pevnou hodnotu.

2. Konečný automat (V, v_0, f), kde V je množina *celulárních stavů*, v_0 je rozlišitelný element množiny V nazývaný jako *quiscent stav* (klidový stav) a f je lokální přechodová funkce *n-tic* z V do V $(f: V^n \rightarrow V)$. Funkce f má následující omezení:

$$f(v_0, v_0, \dots, v_0) = v_0. \tag{2.2}$$

V podstatě máme (2-dimenzionální) mřížku propojených buněk, kde každá buňka obsahuje identickou kopii konečného automatu (V, v_0, f) (Poznámka: Toto platí pro uniformní variantu CA, pro neuniformní může obsahovat každá buňka jinou přechodovou funkci. Jinými slovy f závisí na α, tj. f_α. Problematiku uniformní vs. neuniformní CA probírá následující kapitola). Stav $v^t(\alpha)$ buňky α v čase t je přesně stavem přidruženého konečného automatu v čase t. Každá buňka α je spojena s n sousedními buňkami $\alpha + \delta_1, \alpha + \delta_2, \dots, \alpha + \delta_n$. V následujícím textu předpokládáme, že jeden ze sousedů α je sama α a zavedeme konvenci $\delta_1 = (0, 0)$.

Funkce stavu sousedství $h^t: I \times I \rightarrow V^n$ je definována předpisem

$$h^t(\alpha) = \left(v^t(\alpha), v^t(\alpha + \delta_2), \dots, v^t(\alpha + \delta_n)\right). \tag{2.3}$$

Nyní můžeme dát do relace stav sousedství buňky α v čase t s celulárním stavem buňky v čase $t + 1$ zápisem

$$f\left(h^t(\alpha)\right) = v^{t+1}(\alpha). \tag{2.4}$$

Funkce f je označována jako *pravidlo* (*rule*) CA a bývá obvykle zapisováno ve formě *tabulky pravidel* (*rule table*), která specifikuje všechny možné páry tvaru $\left(h^t(\alpha), v^{t+1}(\alpha)\right)$. Takovýto pár pak označujeme pojmy *transition* (*přechod*) nebo *rule-table entry*.

Konfigurací nazýváme přípustné přiřazení stavů všem buňkám CA. Tudíž, konfigurace je funkce $c: I \times I \rightarrow V$ taková, že množina

$$\{\alpha \in I \times I \mid c(\alpha) \neq v_0\} \tag{2.5}$$

je konečná. Říkáme, že tato funkce má *konečný nosič funkce v relaci k* v_0. Jelikož $f(v_0, v_0, \dots, v_0) = v_0$ (tj. že buňka, jejíž sousedství je v klidovém stavu, zůstává v klidovém stavu i nadále), tak v každém časovém kroku jsou všechny buňky s výjimkou jistého konečného počtu buněk v klidovém stavu.

Nyní si můžeme nadefinovat *globální přechodovou funkci* F. Nechť C je třída všech konfigurací v zadaném celulárním prostoru. Pak $F: C \to C$ je definována předpisem

$$F(c)(\alpha) = f\big(h(\alpha)\big) \tag{2.6}$$

pro všechny $\alpha \in I \times I$. Zadáním nějaké počáteční konfigurace c_0, pak funkce F rozlišuje sekvenci konfigurací

$$c_0, c_1, \dots, c_t, \dots \tag{2.7}$$

kde

$$c_{t+1} = F(c_t) \tag{2.8}$$

pro všechna t.

2.3 Uniformní a neuniformní CA

Jak již bylo uvedeno výše, tak celulární automaty obsahují lokální přechodová pravidla, resp. jejich buňky je obsahují. Tedy obecně má každá buňka svoje vlastní přechodové pravidlo. Takovýto CA nazveme *neuniformním*, příp. *hybridním* [7]. V praxi daleko častějším případem ale je, že všechny buňky mají tato pravidla shodná, tedy existuje pouze jedno přechodové pravidlo pro celý CA. Takovýto CA pak označíme jako *uniformní*.

Podle [1] můžeme definovat ještě jednu třídu celulárních automatů, a to tzv. *kvazi-uniformní* CA. Tento CA je obecně neuniformním CA, kde ale většina buněk obsahuje shodné pravidlo (dominantní pravidlo). Rozlišujeme 2 typy:

- 1. typ – existuje více dominantních pravidel, které pokrývají většinu plochy CA.
- 2. typ – existuje pouze jedno dominantní pravidlo, které pokrývá většinu plochy CA.

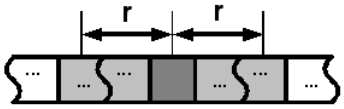
V obou variantách pak pro zbylé buňky (neobsahující dominantní pravidla) platí, že obsahují konečný počet pravidel odlišných od dominantních.

2.4 Typy sousedství

Buňky CA mění svůj stav na základě lokální přechodové funkce (pravidlo). Z kapitoly 2.2 víme, že tato funkce je zobrazení z množiny stavů buňky (pro kterou provádím přechod) a jejích okolních (sousedních) buněk do množiny stavů buňky. Tuto množinu nazýváme jako *sousedství*

(*neighbourhood*). Používá se mnoho typů sousedství, které se liší podle počtu dimenzí CA, velikosti, nebo tvaru.

V 1D CA se asi nejčastěji setkáváme se sousedstvím sestávajícím se z dané buňky a $2 \times r$ bezprostředně sousedících buněk, kde r označuje *rádius*. Rádius udává počet sousedících buněk zleva i zprava, které se zahrnují do sousedství [7]. V případě použití tabulky pravidel (viz. kapitola 2.2), tak pro s stavů a rádius r pak tato tabulka bude obsahovat $s^{2\times r+1}$ položek (pravidel).



Obrázek 2.3: Radiální sousedství (převzato z [6]).

V 2D CA jsou nejpoužívanějšími typy sousedství *Moorovo* a *von Neumannovo*. Označme si množinu buněk tvořící sousedství buňky ležící na souřadnicích (x_0, y_0) v CA M jako $N^M_{(x_0,y_0)}$. [4] Pak pro Moorovo sousedství je tato množina definována následovně:

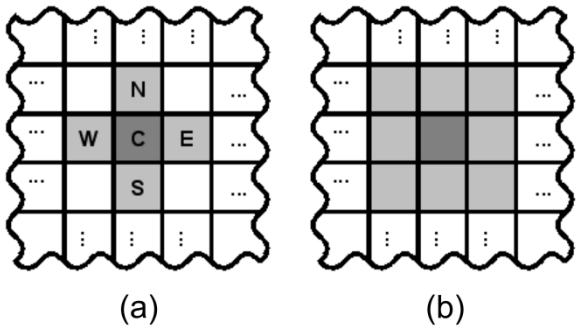
$$N^M_{(x_0,y_0)} = \{(x, y) : |x - x_0| \leq r, |y - y_0| \leq r\}, \qquad (2.9)$$

kde r je rádius. Počet buněk v sousedství lze spočítat podle rovnice $(2r + 1)^2$. Tabulka pravidel bude obsahovat $s^{(2r+1)^2}$ položek.

[5] Pro von Neumannovo sousedství definujeme $N^M_{(x_0,y_0)}$ následovně:

$$N^M_{(x_0,y_0)} = \{(x, y) : |x - x_0| + |y - y_0| \leq r\}, \qquad (2.10)$$

kde r je rádius. Počet buněk v sousedství lze spočítat podle rovnice $2r(r + 1) + 1$. Tabulka pravidel bude obsahovat $s^{2r(r+1)+1}$ položek.



Obrázek 2.4: Typy sousedství buněk v 2D CA: (a) von Neumannovo sousedství, (b) Moorovo sousedství (převzato z [6])

Druhů sousedství je samozřejmě daleko víc, ale nebudu se jimi dále zabývat.

(Poznámka: V této práci používám pro označení sousedství i slovo okolí, tj. Moorovo sousedství = Moorovo okolí, apod.)

2.5 Okrajové podmínky

Protože v praxi nepracujeme s neomezenými celulárními automaty, je třeba definovat *okrajové podmínky*. Neformálně tyto podmínky definují, jakou buňku (resp. stav buňky) vzít, pokud překročíme hranice CA. Podle [8] rozeznáváme několik druhů podmínek. Mezi nejpoužívanější patří následující:

- **Pevné (fixed)** – mimo hranice CA je konstantní hodnota.

- **Periodické (periodic)** – při překročení beru hodnotu z protější strany CA. Známé také jako **cyklické**. Pro 1D CA vytváří kruh, pro 2D vytváří toroid.

Obrázek 2.5: Okrajové podmínky (převzato z [8]).

2.6 Jednorozměrný CA a Wolframovy třídy

Ačkoliv se může zdát, že 1D varianta CA nalezne stěží nějaké využití, není to pravda. Např. se používá k simulaci dopravního toku (*traffic flow*) nebo často řešenou úlohou je tzv. *problém majority* (*majority problem*, nebo *density classification task*). Zde se snažíme nalézt takové pravidlo, u kterého pro nějakou počáteční konfiguraci skončí automat po jistém počtu vývojových kroků v konfiguraci, kdy všechny buňky budou ve stejném stavu, kde tento stav byl v počáteční konfiguraci majoritním, tj. tento stav měl nejvyšší počet zastoupení.

Jednorozměrný uniformní CA také zvolil Stephen Wolfram jako jednoduché počítačové prostředí pro studium otázky, odkud se bere komplexita [2]. Studoval celulární automaty a jejich vztah k dynamickým systémům [1] a identifikoval následující 4 třídy chování [1][2][9]:

1. **Třída 1** – CA této třídy přejde postupem času do pevného, statického, homogenního stavu.

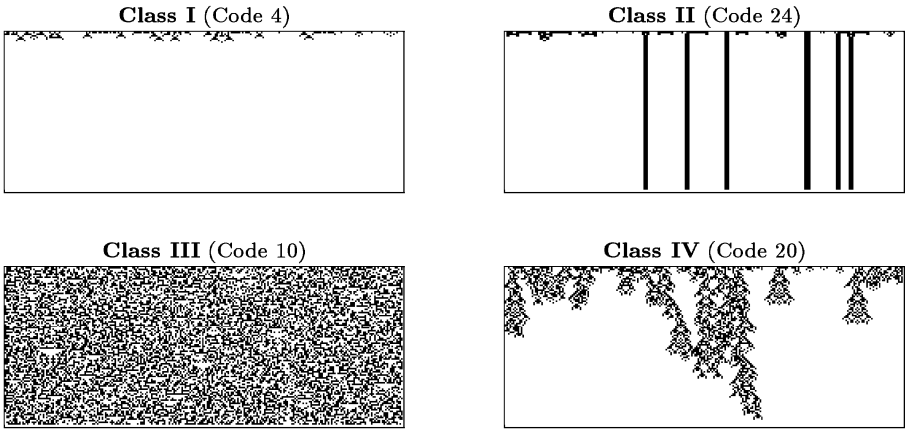
2. **Třída 2** – zde se CA dostane do buď stálé nehomogenní konfigurace nebo se vytvoří struktury, které se donekonečna opakují.

3. **Třída 3** – představuje takové chování, kdy se v CA vytvářejí struktury, které se nikdy neopakují (automat se chová chaoticky), vzor se zdá být náhodným.

4. **Třída 4** – v CA nepravidelně vyrůstají a zase mizí komplexní obrazce.

Wolframova klasifikace nevysvětluje, jak tyto třídy vznikají a jaké jsou mezi nimi vztahy, ale pouze dělí celulární automaty podle toho „jak se nám jeví“.

Poslední třída se jeví jako nejzáhadnější a automaty této třídy projevují nejkomplexnější chování. Wolfram přišel s hypotézou, že celulární automaty **třídy 4** budou obecně schopné univerzálních výpočtů. Později se ukázalo, že **třída 4** tvoří hranici mezi **třídou 2** (periodickým chováním) a **třídou 3** (chaotickým chováním). [2]



Obrázek 2.6: Příklady chování 1D celulárních automatů pro všechny 4 třídy chování [19]. Počáteční stavy automatů jsou zobrazeny na horizontální přímce u vrcholu obdélníků. Vertikálním směrem od shora dolů jsou po řádcích zobrazovány jednotlivé vývojové kroky automatů. V závorce je zobrazen kód použitého totalistického pravidla (o totalistických pravidlech a kódech se dozvíte více v [18]).

2.7 Dvourozměrný CA a Game of Life

Mezi nejvíce používané varianty vícerozměrných celulárních automatů patří bezesporu ty dvourozměrné. U více než 2 rozměrů již nastává problém se značným nárůstem časové a především paměťové náročnosti. Toto lze řešit např. použitím specializovaného hardwaru.

2D (a obecně libovolně rozměrné) CA mají široké možnosti použití. Namátkou uvedu několik příkladů: sebe-replikace struktur, simulace sypání písku, simulace šíření epidemií, simulace proudění tekutin, simulace růstu krystalů, atp. Ale asi nejznámějším příkladem 2D CA je výtvor britského matematika Johna Conwaye. Vymyslel hru zvanou *Game of Life* (nebo krátce *Life*, doslovný překlad je *Hra života*). Hra je to jen v uvozovkách, protože hráč pouze zvolí počáteční konfiguraci CA a provádí vývoj automatu podle daných pravidel. Jedná se o simulaci života, kde pravidla představují zákony fyziky (nebo života) a obrazce jsou materiální objekty. Tento uniformní CA má ortogonální strukturu, 2 stavy (živá/neživá buňka), používá Moorova okolí a obsahuje jednoduchou sadu pravidel [9]:

- Živá buňka se 2 nebo 3 sousedy přežívá.
- Živá buňka s více jako 3 sousedy umírá na přelidnění.
- Živá buňka s méně jako 2 sousedy umírá na osamocení.
- Neživá buňka se 3 živými sousedy ožívá.

Aplikováním základních pravidel může z jednoduchého startovního uspořádání povstat velice komplexní chování. Hru *Life* řadíme ve Wolframově klasifikaci (viz. kapitola 2.6) do třídy 4. Ukázalo se, že na počátku konečná populace může neomezeně růst. Za tento objev vděčíme skupině z MIT, která nalezla strukturu známou jako „kluzákové dělo“ (glider gun), která každých 30 kroků vypouští jeden kluzák (glider). Bylo nalezeno mnoho dalších struktur, které nesou jména jako bochník, rybník, vana, loď, včelín, apod. Obrazce vytvářené hrou můžou být neobyčejně složité. Dá se ukázat, že tento automat je ekvivalentní univerzálnímu Turingovu stroji, tzn. že jakýkoliv výpočet provedený na Turingovu stroji se dá provést i pomocí hry *Life*. [2]

3 Sebe-replikace v celulárních automatech

Sebe-replikace (*self-replication*), nebo také *sebe-reprodukce* (*self-reproduction*) je proces vytváření kopií sama sebe. (Poznámka: Ve většině prací, včetně této práce, zabývajících se tématikou sebe-replikace se tyto termíny považují za synonyma. V [11] mezi nimi dělají rozdíl. Replikaci řadí mezi ontogenetické procesy a reprodukci mezi fylogenetické procesy. Více informací naleznete v [11] nebo v [10].) Za posledních 60 let, kdy je problematika umělých sebe-replikujících se struktur a strojů studována a řešena, prošla značným vývojem od ryze teoretických prací po pokusy o praktické řešení. Velká část těchto prací byla motivována touhou po pochopení fundamentálních informačně procesních principů a algoritmů souvisejících se sebe-replikací, nezávisle na jejich fyzické realizaci [10]. Pochopení těchto principů může být užitečné z mnoha pohledů. Například může pomoci k lepšímu pochopení biologické replikace, objasnit původ života [12]. Výroba umělých sebe-replikujících se strojů pak může mít rozličné uplatnění, od nanotechnologií [13], přes průzkum vesmíru [14], po rekonfigurovatelné výpočetní platformy (angl. reconfigurable computing tissues) [15].

Použití celulárních automatů není jediným přístupem k výzkumu v oblasti sebe-replikace, ale je zdaleka nejpoužívanější [10]. Na tuto oblast se dále zaměřuji. Mezi další přístupy patří například *sebe-replikující se programy* (*self-replicating programs*), *sebe-replikující se řetězce* (*self-replicating strings/strands*) a další [10][17].

Jak už jsem výše uvedl, tak často používaným výpočetním modelem pro výzkum sebe-replikujících se struktur či strojů jsou celulární automaty. V [16] rozdělili tyto modely do tří skupin podle přístupu k řešení:

- První sebe-replikující se struktury (60. a 70. léta minulého století) byly veliké a komplikované univerzální systémy modelované podle Turingova stroje. Ty nám ukázaly, že umělé sebe-replikace struktur lze dosáhnout a tím stimulovaly následné důležité teoretické práce.
- Druhou generací sebe-replikujících struktur studovaných od poloviny 80. let minulého století jsou *sebe-replikující se smyčky*. Byly navrhovány tak, aby byly kvalitativně jednodušší než jejich předchůdci, čehož bylo dosaženo zmírněním požadavků na výpočetní univerzálnost a konstrukci.
- Poslední a nejnovější přístupy se oproti dřívějšímu přístupu ve formě manuálního návrhu replikátorů zaměřují na sebe-replikaci více jako na *emergentní vlastnost*. Sebe-replikující se struktury můžou „vyplynout“ z náhodného počátečního stavu za použití umělých evolučních metod (např. genetické algoritmy). Bylo také ukázáno, že tyto struktury můžou být použity k řešení problémů (např. SAT problém) během jejich činnosti.

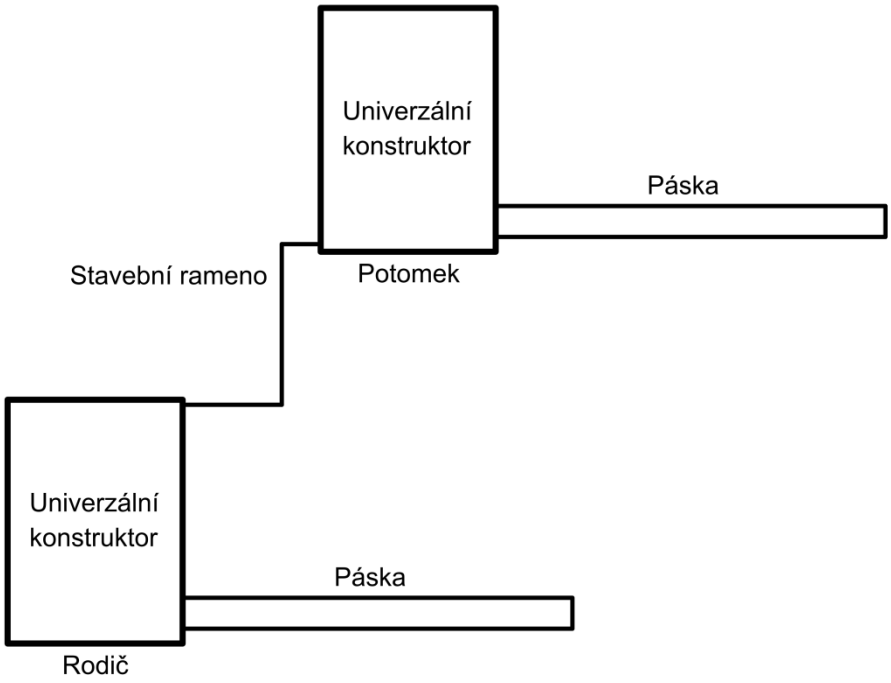
3.1 Von Neumannův univerzální konstruktor

V úvodu 2. kapitoly jsem se zmínil o von Neumannovi, který se zabýval problémem sebe-replikace a pro ten účel si vytvořil výpočetní model plovoucího robota. Ale více se mu zalíbil Ulamův celulární

automat, který následně použil a dal tak vzniknout tomu, co dnes známe pod jménem *von Neumannův univerzální konstruktor* (*universal constructor*, nebo také *universal computer-constructor*).

K řešení tohoto problému si zvolil dvourozměrný CA s 29 stavy na buňku a 5ti buňkové okolí (dnes známé jako von Neumannovo, viz. kapitola 2.4). Sebe-replikující se objekt se skládal z cca 200 000 buněk, které tvořily tělo sebe-replikující se struktury, stavební rameno (*constructing arm*) a jakýsi ocas, který představuje vstupní pásku (*tape*). Tento automat pak dokáže pomocí stavebního ramene vytvořit kopii sama sebe podle instrukcí uložených na pásce. Jméno *univerzální konstruktor* dostal proto, že dokáže zkonstruovat libovolnou strukturu uloženou na pásce. Tuto pásku dokáže také zkopírovat do nově vytvořené struktury. O sebe-replikaci pak hovoříme v případě, že vložíme na pásku popis vlastní struktury. Později se podařilo tento automat výrazně zjednodušit. [2][10][16]

Tato práce ukázala, že v celulárních automatech lze sestrojit univerzální výpočetní stroj, jehož výpočetní síla je rovna výpočetní síle Turingova stroje [10]. Dále ukázala, jaký druh komplexity vyžaduje sebe-replikace. Tyto automaty se pak můžou vyvíjet darwinovským způsobem. Nenašel se žádný nezvratný důkaz o zásadní propasti mezi člověkem a strojem. [2]



Obrázek 3.1: Schematický diagram von Neumannova sebe-replikujícího se celulárního automatu.

3.2 Coddův celulární automat

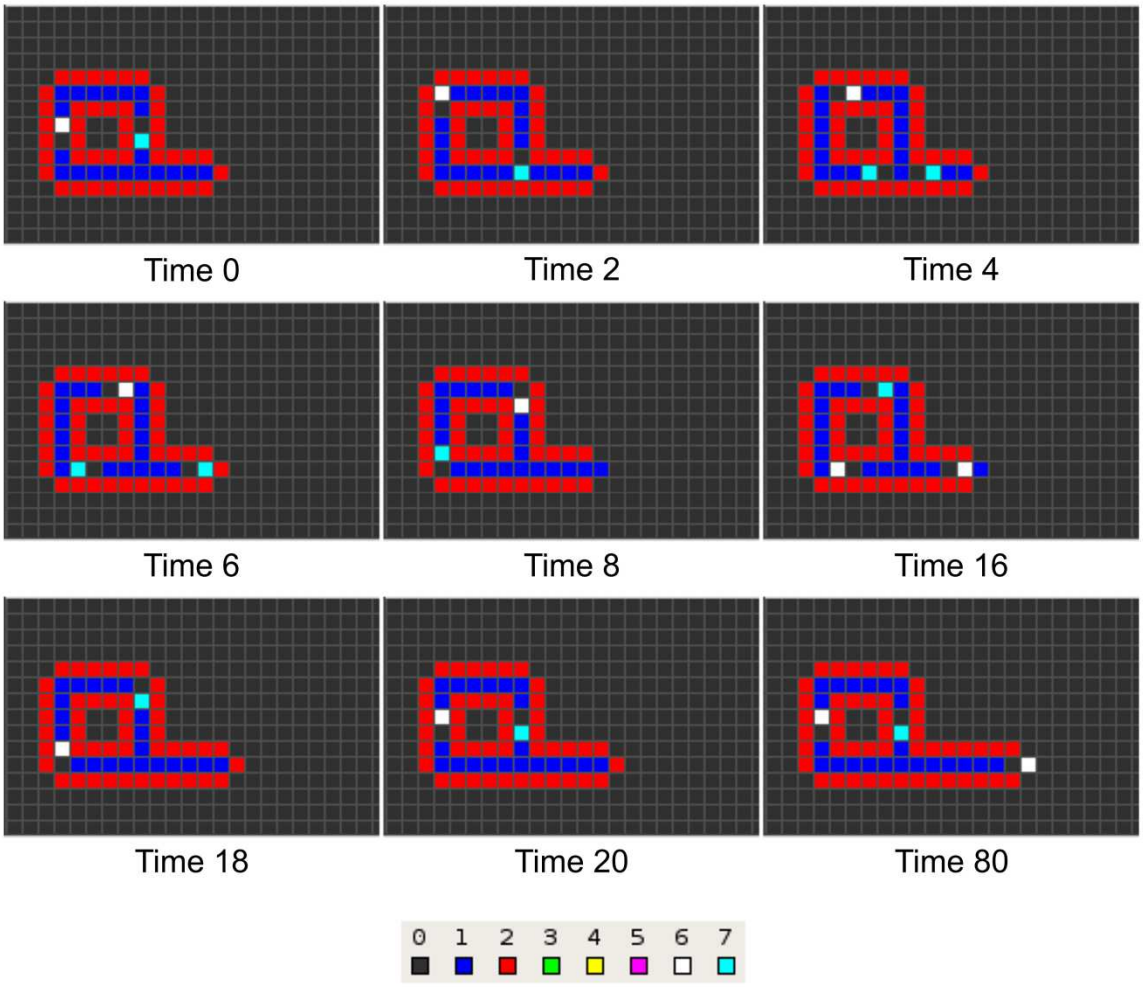
V kapitole 3.1 jsem zmínil, že se podařilo von Neumannův univerzální konstruktor zjednodušit. Jedním z těch, kdo se tímto problémem zabýval, byl britský informatik Edgar F. Codd. Kromě důležitých prací v oblasti databázových systémů tak vytvořil v roce 1968 v rámci svého doktorského výzkumu celulární automat, ve kterém zredukoval komplexitu von Neumannova automatu [20].

V tomto svém automatu provedl redukci původních 29 stavů na 8 a sebe-replikující struktura pak obsahuje kolem 100 000 000 buněk [10]. Zanechal von Neumannovo okolí. Automat je schopen univerzální konstrukce a sebe-replikace je dosaženo jako speciálního případu univerzální konstrukce, stejně jako v práci von Neumanna. Oba stroje se chovají vlastně velice podobně, rozdíl je vidět

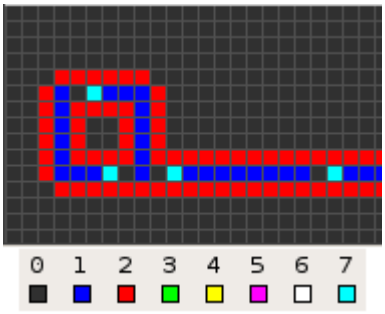
především v konstrukci, kde Codd byl ovlivněn fyziologií nervového systému zvířat. A i když je Coddův automat jednodušší nežli von Neumannův, stále patří mezi velice komplexní stroje, jako jsou moderní počítače. [20]

3.2.1 Coddova smyčka

Kapitola vychází z [20] a [21]. *Periodic emitter* je název jedné ze struktur, které Codd vytvořil ve svém automatu a použil ji jako základní a důležitý časovací prvek v jeho sebe-replikujícím se stroji. Jedná se o jednoduchou nereplikující se strukturu tvaru smyčky s ramenem, ve které neustále cirkuluje sekvence signálů. Vždy, když míjí tato sekvence rameno, tak se propaguje také do něj, což si můžeme představit jako tiknutí hodin ve stroji.



Obrázek 3.2: Ukázka vývoje smyčky s ramenem se dvěma signály v Coddově celulárním automatu. Stav 0 značí klidový stav, stav 1 *jádro* (*core*), stav 2 *obal* (*sheath*). Signály jsou složeny ze 2 stavů, tj. stavy 6 a 7 cestují spolu se stavem 0. Sekvence signálů 7 0 – 6 0 po dosáhnutí konce ramena vyústí v jeho prodloužení o jednu buňku. [20]



Obrázek 3.3: Příklad Coddova *periodic emitteru*.

Na obrázku 3.2 je ukázka vývoje smyčky s ramenem, ve které cirkuluje sekvence dvou signálů 7 0 a 6 0, které způsobují růst ramene. U složitějších sekvencí signálů může kromě růstu dojít i k jeho zahnutí. Obrázek 3.3 pak ukazuje příklad samotného *periodic emitteru*.

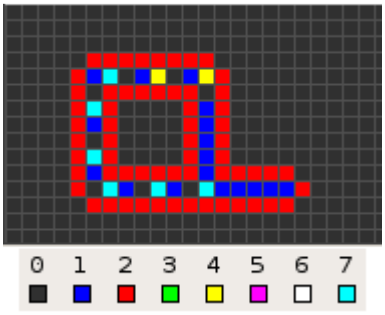
Tato struktura, i když ne sebe-replikující se, tak se stala inspirací pro práci Ch. Langtona a jeho známou sebe-replikující se smyčku (viz. kapitola 3.3.1).

3.3 Sebe-replikující se smyčky

Doposud jsme se bavili o komplexních sebe-replikujících se strojích nebo automatech složených ze statisíců buněk, které je problematické vůbec zkonstruovat a následně provést simulaci na počítačích nebo na jiných výpočetních platformách. V dnešní době se to již nezdá být natolik překážkou, ale v 80. letech minulého století se to jistě jevilo jako značný problém. V té době vzniká první *sebe-replikující se smyčka* (*self-replicating loop*), která inspirovala jiné tvůrce k vzniku dalších a lepších smyček.

3.3.1 Langtonova smyčka

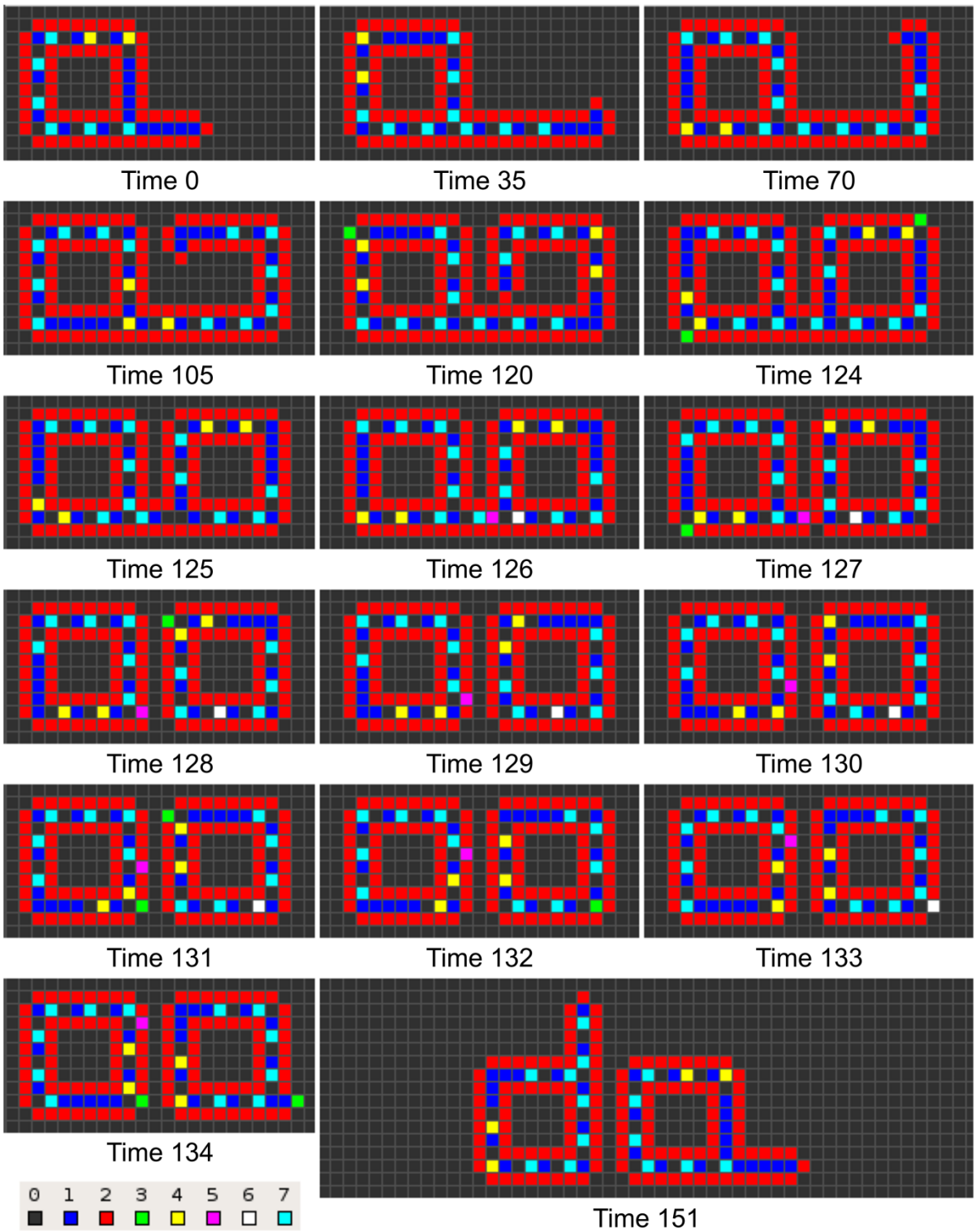
Tato kapitola vychází z [10] a [20]. Langton svoji smyčku publikovanou roku 1984 nazval přesněji *self-reproducing loop*, ale v této práci beru oba termíny jako synonyma (viz. poznámka v úvodu kapitoly 3). Jak již jsem psal výše, tak vycházel z Coddova automatu, konkrétněji ze struktury nazvané *periodic emitter* (viz. kapitola 3.2.1). Jeho automat pracoval se stejným počtem stavů i se stejným okolím, tj. 8 stavů, von Neumannovo okolí. Sebe-replikující se smyčka pak je tvořena z 86 buněk, které tvoří podobnou strukturu jako *periodic emitter*, tedy je založená na *datové cestě* (*data-path*) tvořené stavy 1 (*jádro*, angl. *core*) a 2 (*obal*, angl. *sheath*), ve které proudí data ve formě signálů tvořených zbylými stavy (vyjma stavu 0, který představuje klidový stav, viz. kapitola 2.2). Signály cestují v datové cestě v dvojici se stavem 0. Přesněji jsou stavem 0 následovány.



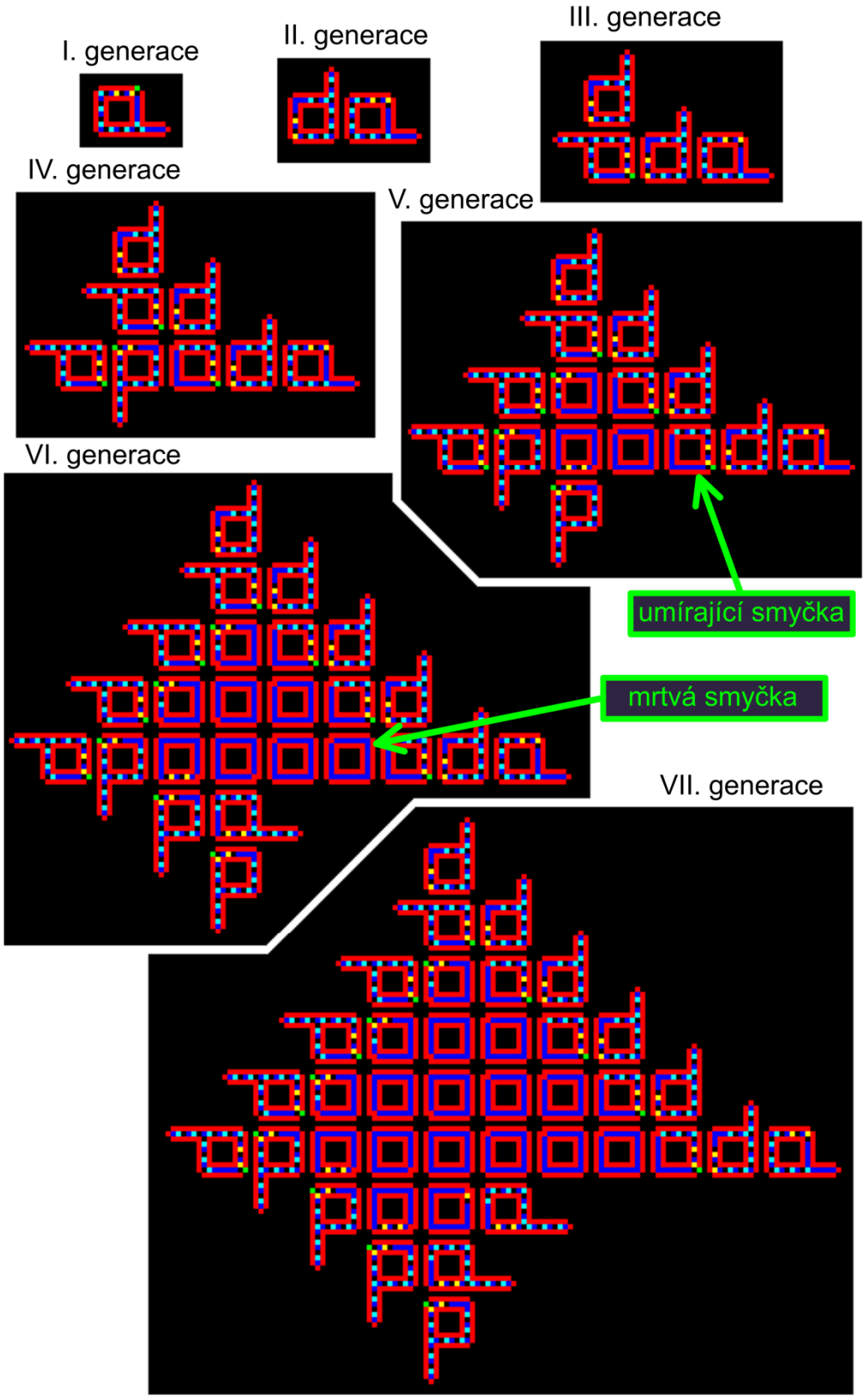
Obrázek 3.4: Langtonova sebe-replikující se smyčka.

Signály cirkulují ve smyčce proti směru hodinových ručiček. Sekvence signálů pak tvoří instrukce pro sebe-replikaci smyčky a představují jakýsi *umělý genom*. Sekvence způsobující sebe-replikaci je:

7 0 - 7 0 - 7 0 - 7 0 - 7 0 - 7 0 - 4 0 - 4 0



Obrázek 3.5: Ukázka několika vývojových kroků Langtonovy sebe-replikující se smyčky od počáteční konfigurace po jednu replikaci.



Obrázek 3.6: Kolonie smyček. Ukázka růstu 7 generací smyček.

Pro tuto sekvenci je smyčka jednou replikována v 151 vývojových krocích. Tohoto chování Langton dosáhl za použití několika stovek pravidel [26] (Poznámka: Přesný počet pravidel je závislý na tom,

zdali jsou použita defaultní pravidla nebo jsou započítána i symetrická pravidla.) z 32 768 možných (viz. kapitola 2.4).

Stejně jako u Coddovy smyčky dochází k duplikaci signálu do ramene, pokud ho míjí. Signály pak postupují ramenem a při dosáhnutí jeho konce jsou pak přeloženy na instrukce způsobující růst ramene, nebo ohnutí ramene. V konečném důsledku dojde k replikaci smyčky i s jejím umělým genomem, takže nově vzniklá smyčka se může začít znovu replikovat stejným způsobem.

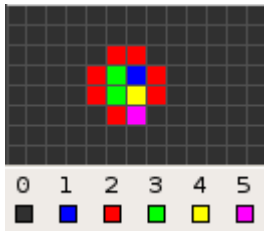
Po ukončení replikace smyčka, která byla replikována, otáčí rameno o 90 stupňů proti směru hodinových ručiček a další kopie bude vytvořena v tomto směru. Tento proces nemůže trvat donekonečna. Replikace probíhá pouze do doby, dokud není replikovaná smyčka obklopena jinými smyčkami. Pokud k tomu dojde, tak se vytvoří v datové cestě blokáda z obalu (stav 2), která způsobí, že se začnou cirkulující signály mazat jeden po druhém (*umírající smyčka*, angl. *dying loop*), až zmizí úplně (*mrtvá smyčka*, angl. *dead loop*). Skupiny smyček nazýváme *koloniemi* (*colony*). Tyto kolonie rostou ve spirálovitém vzoru. Ukázku můžeme vidět na obrázku 3.6.

Langton, stejně jako von Neumann, zdůraznil, že automat musí zacházet s uloženou informací ve smyčce dvěma rozdílnými způsoby: *interpretovaným* (*interpreted*), jako instrukce, která má být provedena (*translace*) a *neinterpretovaným* (*uninterpreted*), jako data, která mají být zkopírována (*transkripce*). Transkripce je dosaženo duplikací signálů na spoji mezi smyčkou a ramenem a translace při překladu signálů na instrukce při dosažení konce ramene, nebo při kolizi s jiným signálem. (Poznámka: Pojmy z molekulární biologie translace a transkripce jsou vysvětleny například v [22].)

3.3.2 Bylova smyčka

Kapitola vychází z [23] a [24]. V článku z roku 1988 (publikovaného v roce 1989 [24]) představil matematik John Byl svoji sebe-replikující se smyčku. Jeho práce vychází z Langtonovy smyčky a jedná se o její zjednodušení, kdy ubral v automatu 2 stavy, zmenšil sebe-replikující se strukturu z 86 buněk na pouhých 12 buněk a zredukoval množinu přechodových pravidel. Zanechal von Neumannovo okolí.

Zjednodušení se skládá z několika modifikací. První modifikace spočívá v odstranění vnitřní stěny obalu. Druhá modifikace je použití pouze signálu 4 0 k provedení otáčky vlevo. S tím souvisí změna ve způsobu určování směru putování signálu ve smyčce. V Langtonově smyčce je to určeno pořadím signálů, tj. první jde signál 4 nebo 7 a je následován stavem 0. V Bylově smyčce je směr určen orientací každé buňky s ohledem na buňky tvořící vnější obal. Poslední modifikací je zkombinování funkcí signálů 5 a 6 do jednoho stavu.

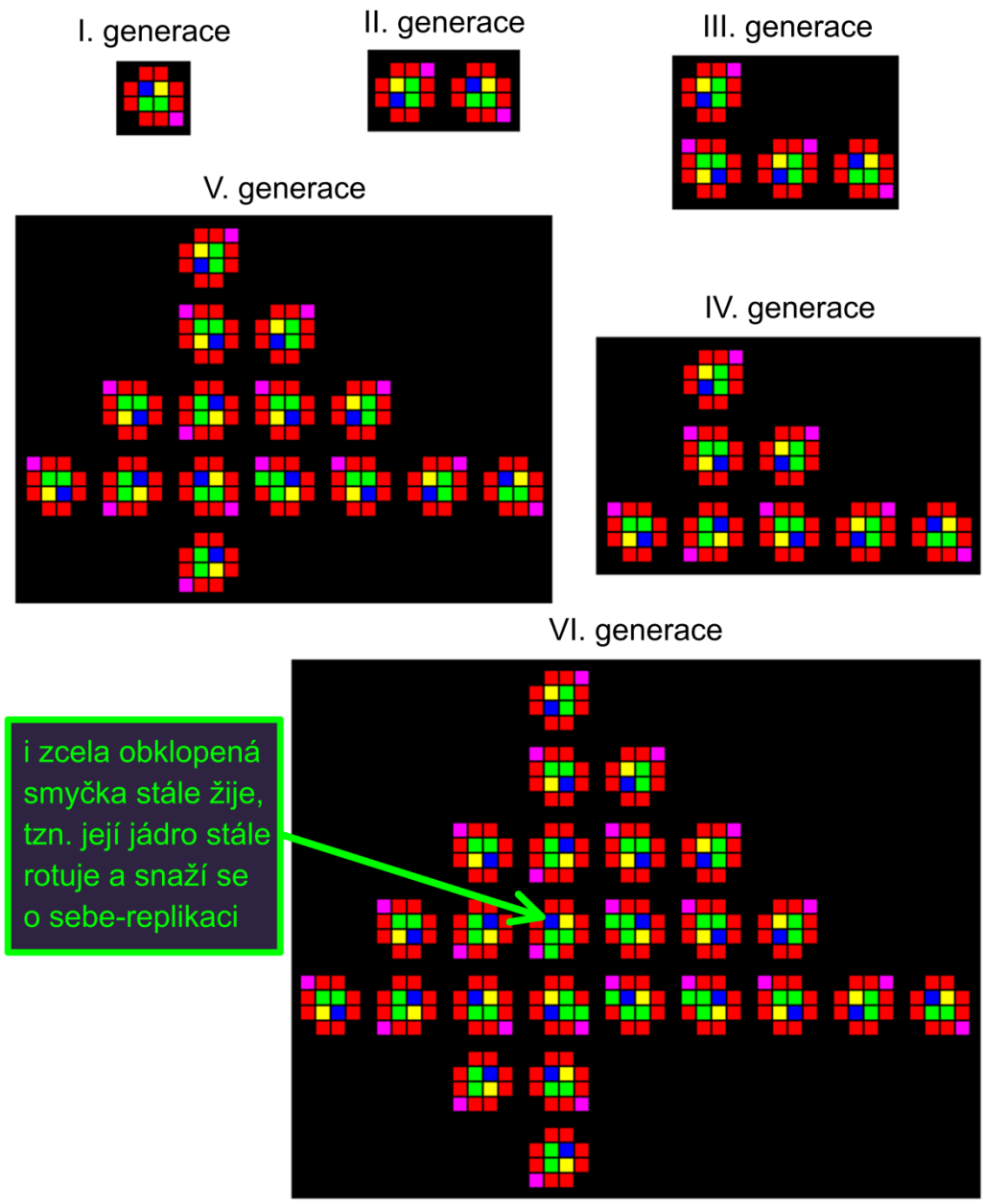


Obrázek 3.7: Bylova sebe-replikující se smyčka.

Sebe-replikace je v Bylově smyčce dosaženo v 25 krocích. Stejně jako u Langtonovy smyčky, tak kopie začne vytvářet svoji kopii napravo od svojí pozice a rodič se natočí o 90 stupňů proti směru hodinových ručiček a zahájí zde sebe-replikaci. A stejně jako u Langtona, tak i zde probíhá sebe-

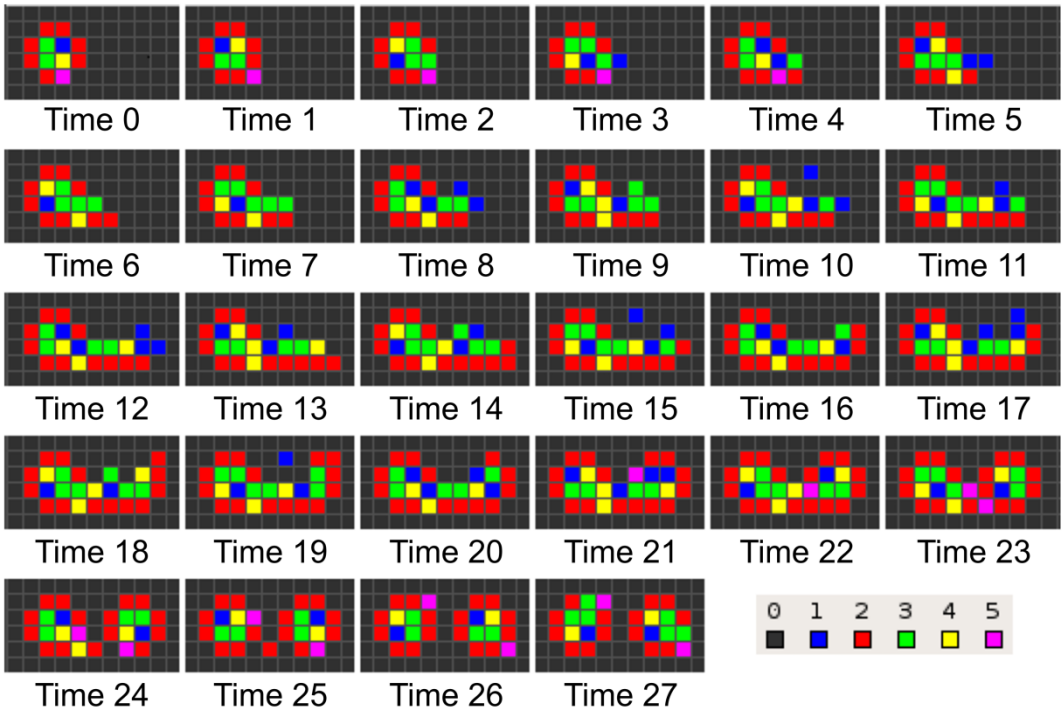
replikace do doby, dokud je volný prostor. Naopak rozdílem je, že smyčka obklopená jinými smyčkami neumírá, ale její jádro neustále rotuje a snaží se sebe-replikovat. Sekvence signálů způsobující sebe-replikaci je:

3 - 3 - 4



Obrázek 3.8: Kolonie smyček. Ukázka růstu prvních 6 generací + jeden vývojový krok.

V článku [23] Byl představuje ještě jinou verzi smyčky. Sestává ze stejného počtu buněk, tj. 12, replikuje se stále v 25 krocích, ale používá 7 stavů místo 6. Více informací naleznete ve zmíněném článku.



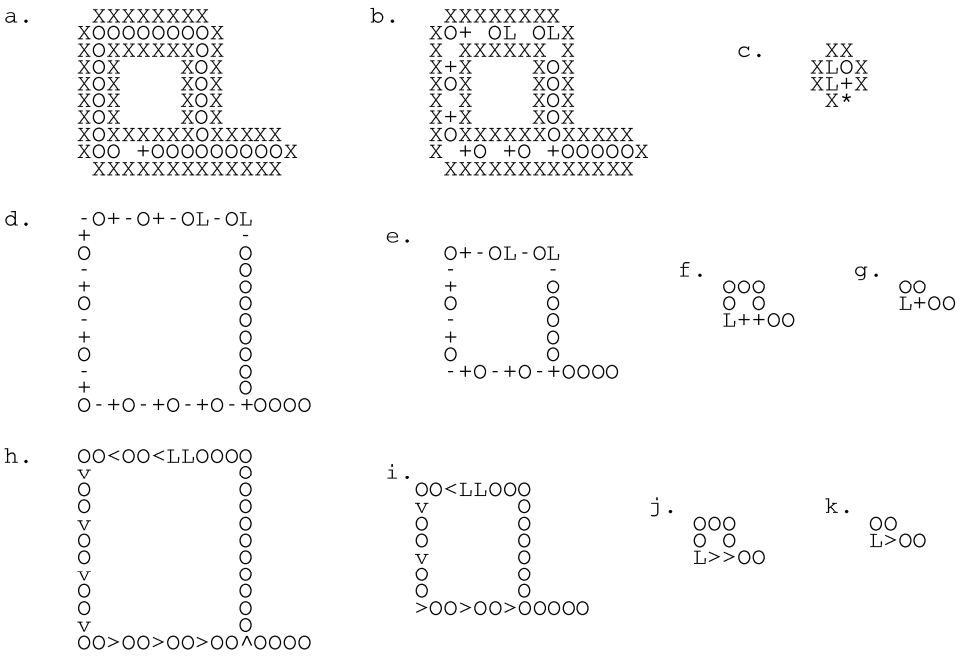
Obrázek 3.9: Ukázka vývoje Bylovy sebe-replikující se smyčky v prvních 27 krocích. V 25. kroku je vytvořena přesná kopie rodičovské smyčky.

3.3.3 Smyčky Choua a Reggia

V [25] z roku 1993 se Reggia, Chou a kol. zabývali myšlenkami na další zjednodušení (Poznámka: U sebe-replikujících se smyček je složitost posuzována na základě počtu stavů automatu, velikosti replikované struktury a počtu použitých přechodových pravidel.) sebe-replikujících se smyček. Po studiu Coddovy práce přišli s hypotézou, že se dají smyčky ještě zjednodušit odstraněním obalu. Dále pak studovali, jak se projevuje u smyček *silná* (*strong*) a *slabá* (*weak*) *rotační symetrie* (*rotational symmetry*) buněk. Silná rotační symetrie znamená, že stavy buněk nemají žádnou směrovou orientaci (jsou rotačně symetrické). Slabá rotační symetrie značí, že některé stavy buněk jsou směrově orientované (např. stav buňky pohybující se nějakým směrem při změně směru změní také svůj stav, tj. pro každý směr má svůj specifický stav). Předchozí smyčky, od Coddovy po Bylovu mají oproti von Neumannovu automatu silnou rotační symetrii.

Vytvořili vlastní verze Coddovy, Langtonovy a Bylovy smyčky, s odstraněným obalem a se silnou i slabou rotační symetrií. Odstraněním obalu dosáhli značného zredukování velikosti struktur a také redukce počtu přechodových pravidel. V případě Bylovy smyčky dosáhnuli i zrychlení replikace, kdy pro silnou rotační symetrii (obr. 3.10g) dojde k replikaci v 14 krocích a pro slabou rotační symetrii (obr. 3.10k) v 10 krocích. Tyto smyčky pracují s von Neumannovým okolím a mají 8 stavů na buňku.

Vytvořil ještě další 2 smyčky se silnou rotační symetrií, které mají stejnou počáteční konfiguraci jako struktura na obr. 3.10 g, kde jedna je tvořena 6 buňkami a druhé je z ramena ubrána jedna buňka (tj. je tvořena pouze 5ti buňkami), pracují pouze s 6 stavy a dokážou se replikovat v 18, resp. 17 krocích.



Obrázek 3.10: Přehled smyček. (a) Coddova nereplikující se smyčka, (b) Langtonova smyčka, (c) Bylova smyčka, (d) až (g) sebe-replikující se smyčky vytvořené pány Chou, Reggia a spol., bez obalu a se silnou rotační symetrií, (h) až (k) podobně jako předchozí 4 smyčky, ale se slabou rotační symetrií (obrázek převzat z [16])

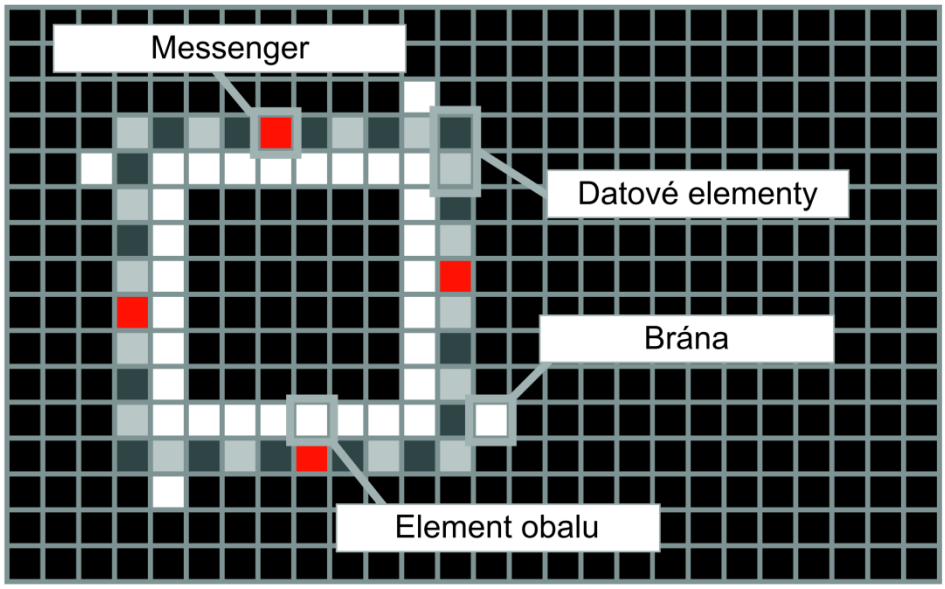
3.3.4 Tempestiho smyčka

V roce 1995 v [27] Ital Gianluca Tempesti představil sebe-replikující se celulární automat schopný navíc konstrukce a výpočtů. Nejedná se tedy o další zjednodušení, ale je to značně rozlišný automat, který je obdařen některými vlastnostmi Langtonovy (Coddovy) smyčky a von Neumannova konceptu konstrukčního ramene.

Jeho smyčka vycházející z Langtonovy smyčky má v základní verzi (bez konstrukčních rozšíření) 5 stavů a používá Moorovo okolí. Ubral ze smyčky vnější obal (Byl ubral ze smyčky naopak ten vnitřní, viz. kapitola 3.3.2), čímž dosáhnul toho, že není potřeba u dat (stavy mimo klidový, ten co tvoří obal a ty co se starají o sebe-replikaci) cirkulujících po smyčce, aby byly sdružené s nějakým vodícím stavem (např. stav 0 u Langtonovy smyčky).

Stejný stav jako má obal mají i „bránové buňky“ (v originále *gate cells*), nebo lépe *brány*, které jsou situovány na vnější stranu smyčky do všech 4 rohů. Tyto buňky můžou být ve dvou pozicích, buď *otevřeno* (*open*), nebo *zavřeno* (*closed*). Pozice otevřeno je počáteční a dovoluje smyčce se replikovat. Po ukončení sebe-replikace se přesune do pozice zavřeno.

Smyčka má místo pouze jednoho konstrukčního ramena hned 4, tzn. že se dokáže replikovat do 4 směrů paralelně. Díky tomu rostou kolonie smyček v symetrickém vzoru (obr. 3.12). Po ukončení replikace dochází k zatažení ramena a přesunu brány do pozice zavřeno. Tyto ramena se navíc prodlužují zcela autonomně (v Langtonově smyčce provádí prodloužení signál 7 0). Důsledkem je problém, kdy zahnout rameno. Toho je dosaženo pomocí zaslání *posla* (*messenger*), který cirkuluje po smyčce a je periodicky posílán do jednotlivých ramen. Aby posel byl schopen dosáhnout konce ramena, tak je redukována rychlost růstu ramena. Konstrukční rameno nevytvoří hned kompletní smyčku, ale pouze její vnitřní obal. Po jeho dokončení se data z rodičovské smyčky duplikují a jsou poslána podél ramena do nové smyčky.

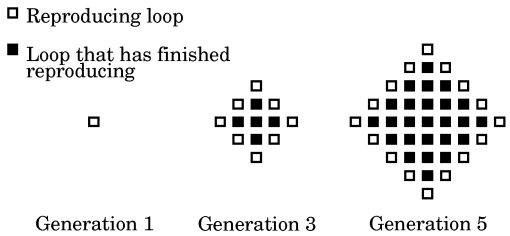


Obrázek 3.11: Tempestiho smyčka (obrázek převzat z [26]).

V důsledku nevyužití datových buněk ke konstrukci ramena (pouze 4 buňky jsou použity k natočení), tak zbylé datové buňky nemusí být v žádném určitém stavu a můžou být použity jako „program“, tj. mají vlastní množinu stavů a vlastní přechodová pravidla, která jsou aplikována společně se sebe-replikací. Takovýto program je spuštěn v rodičovské smyčce a posléze v jejích všech kopiích, atd. (ukázka na obr. 3.14).

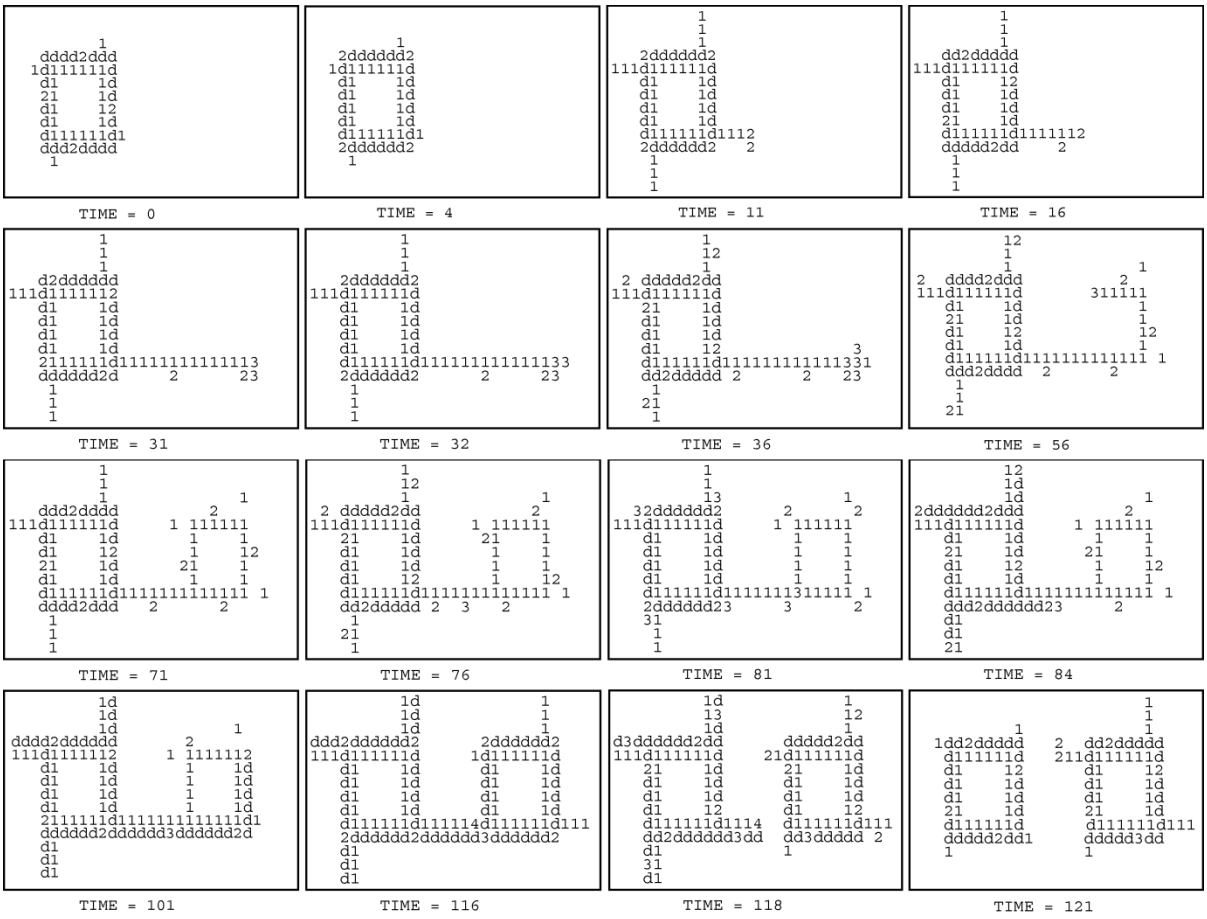
Oproti Langtonově smyčce, Tempestiho smyčka neumírá po dokončení sebe-replikace, tj. data jsou nedotčena a neustále cirkulují, takže jakýkoliv program uložený ve smyčce je stále možno spustit. Také se dá znovu aktivovat sebe-replikační proces pouhým přesunutím brány do pozice otevřeno.

Pokud konstrukční rameno při růstu narazí na okraj celulárního automatu nebo na jinou smyčku, tak se zatáhne bez pokusu o duplikaci dat. Langtonova smyčka při nárazu na okraj havaruje (angl. crash).

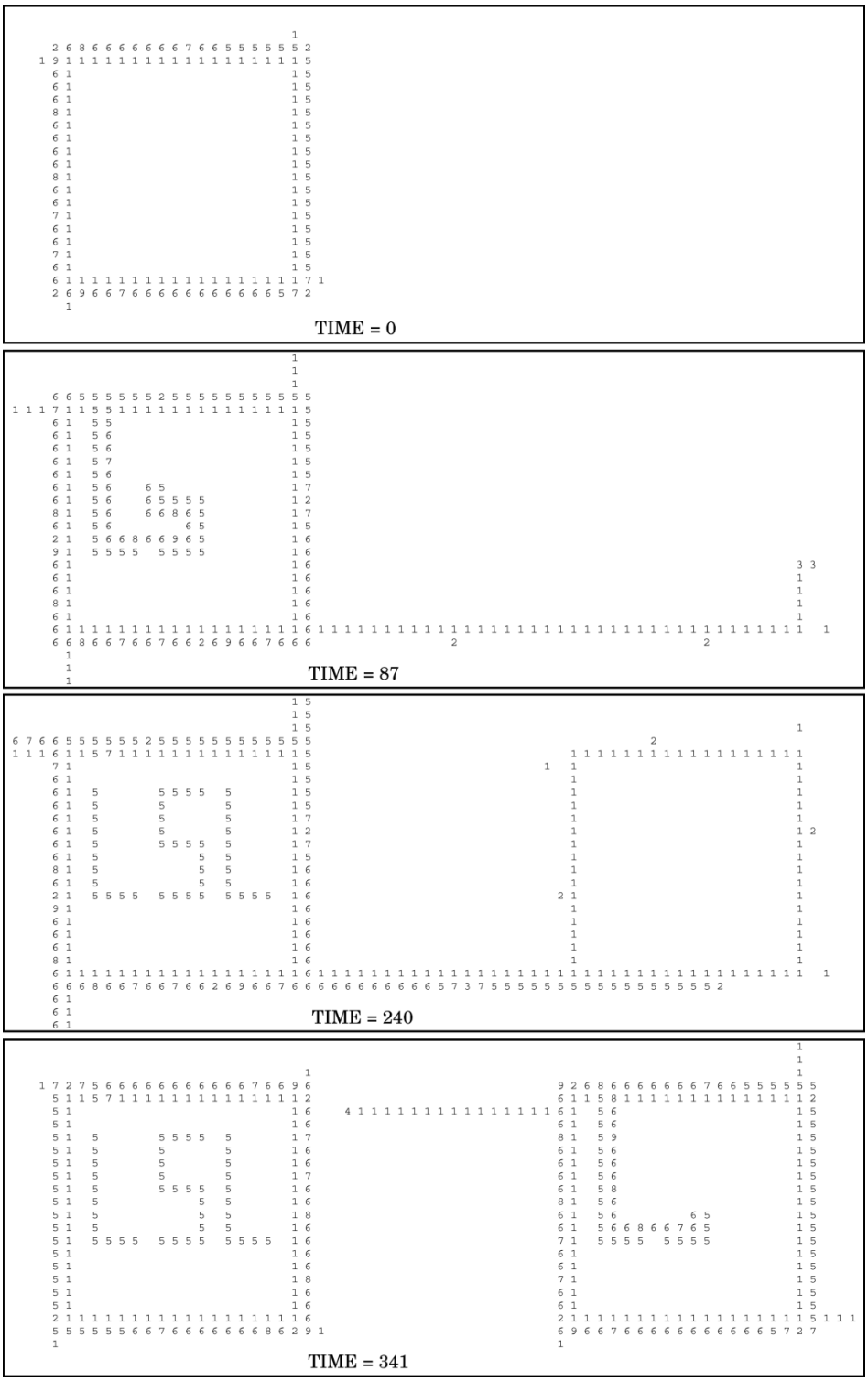


Obrázek 3.12: Symetrický růstový vzor Tempestiho smyčky. (Obrázek převzat z [27].)

Co se týče komplexity automatu, potažmo velikosti smyčky, tak ta je závislá na množství datových buněk, resp. složitosti (délky) cirkulujícího programu. Tedy, čím složitější program, tím větší musí být smyčka. Ale komplexita sebe-reprodukčního procesu není žádným způsobem závislá na velikosti smyčky.



Obrázek 3.13: Ukázka několika vývojových kroků Tempestiho sebe-replikující se smyčky od počáteční konfigurace po jednu replikaci. (Obrázek je vytvořen ze zdrojů v [27].)



Obrázek 3.14: Příklad konstrukčních schopností Tempestiho automatu. (Obrázek převzat z [27].)

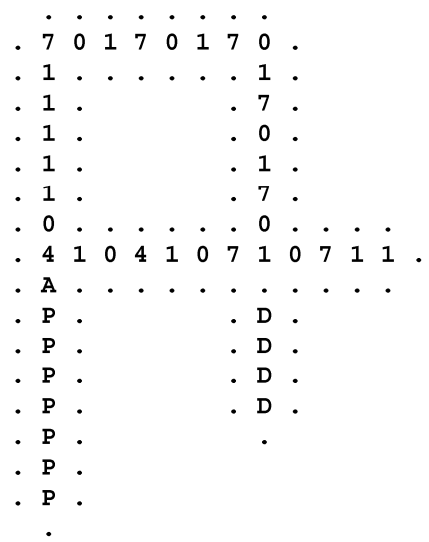
3.3.5 Perrierova smyčka

V článku [28] z roku 1996 Perrier a kol. představují další sebe-replikující se smyčku, u které bylo cílem jejího vzniku schopnost univerzálního výpočtu a zároveň aby byla fyzicky realizovatelná (narážka na von Neumannův automat, který bylo v době vzniku smyčky prakticky nemožné s tehdejší technikou realizovat). Jedná se v podstatě o Langtonovu smyčku, které byly přidány schopnosti

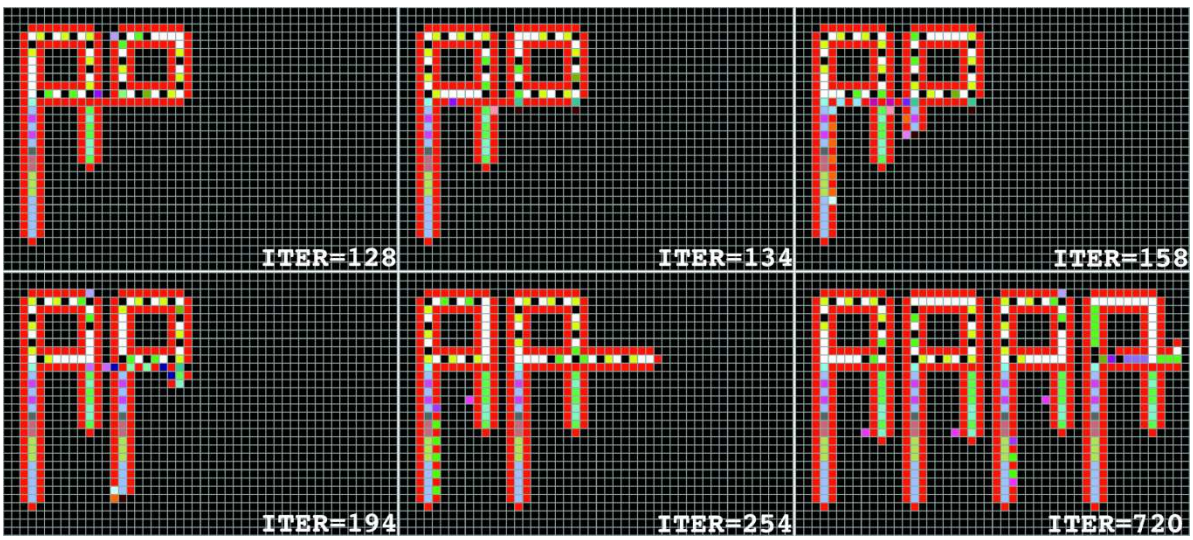
univerzálního výpočtu. Tato schopnost jí byla přidána ve formě Turingova stroje, konkrétně jeho modelu představeného Wangem v [29] a nazvaném později *W-machine*.

Program a data pro *W-machine* jsou uloženy externě mimo smyčku ve dvou páskách majících lineární strukturu. Páska s programem má již z principu pevnou velikost, oproti pásce s daty, která má jedním směrem nekonečnou velikost.

Perrierova smyčka má silnou rotační symetrii a používá dva typy signálů. Prvním typem je Coddova verze signálů, které jsou použity k zachování genomu smyčky, tj. instrukce potřebné k její sebe-replikaci. Druhým typem je Langtonova zjednodušená verze signálů, která jsou použity k manipulaci s informacemi souvisejícími s programem a daty.



Obrázek 3.15: Struktura Perrierova automatu. *P* označuje stav patřící do množiny stavů programu, *D* označuje stav patřící do množiny stavů dat, *A* je stav označující pozici v programu (obrázek je převzat z [28])



Obrázek 3.16: Sebe-replikace Perrierovy smyčky. (Obrázek převzat z [26].)

Při sebe-replikaci nejprve dojde k replikaci samotné smyčky, po jejímž ukončení zůstanou obě smyčky, tj. rodič a potomek spojeni stále pomocí „pupeční šňůry“ (umbilical cord; pojem zavedený Langtonem pro rameno spojující rodičovskou smyčku a potomka). Po této pupeční šňůře se pak

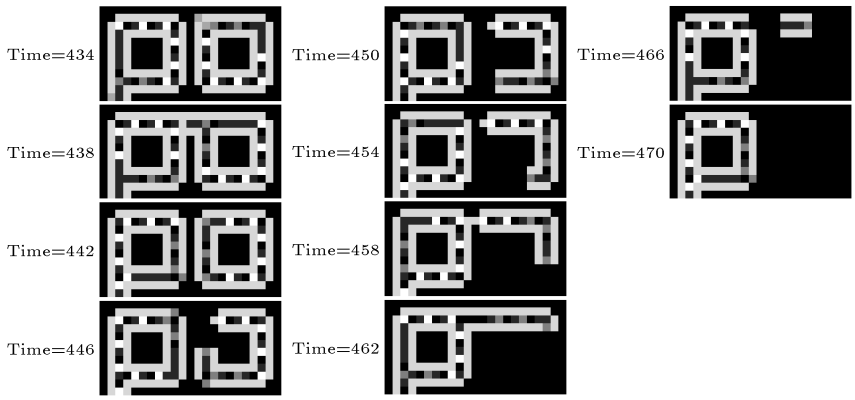
provede reprodukce programové pásky z rodiče do potomka a následně i reprodukce datové pásky. Po ukončení sebe-replikace dojde k přerušení pupeční šňůry a spustí se provádění programu v rodičovské smyčce. Bližší informace k provádění programu a k sebe-replikaci naleznete v [28].

3.3.6 SDSR smyčka

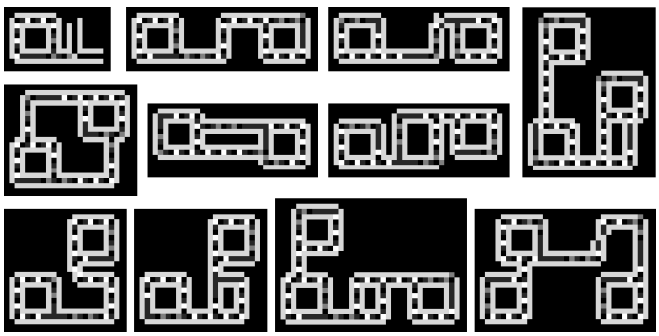
Hiroki Sayama se v roce 1998 v [30] zamýšlel nad otázkami „Co je život?“ a „Jak se živý systém chová?“ v souvislosti se studiem Langtonovy sebe-replikující se smyčky (v práci zkracuje jako SR smyčka, angl. SR loop). Langtonova smyčka pokud již nemá prostor k sebe-replikaci, tak umírá (viz. kap. 3.3.1). Pro toto úmrtí použil termín *smrt jako funkční selhání* (*death as functional failure*). Ve skutečném životě ale s koncem života přichází také rozklad fyzické struktury systému. Toto označil termínem *smrt jako strukturní rozklad* (*death as structural dissolution*). Výstupem jeho práce je nová smyčka nazvaná jako *SDSR smyčka* (*SDSR loop*; *structurally dissolvable self-reproducing loop*).

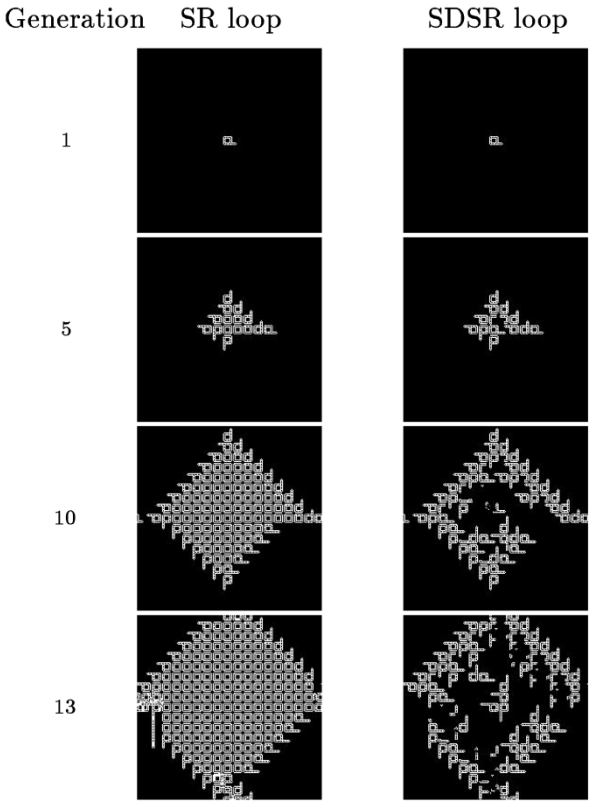
Rozšířil Langtonových původních 8 stavů na 9 zavedením *rozkládacího stavu* (*dissolving state*). Tento stav v případě úmrtí smyčky zajistí její rozklad a tím uvolní místo pro další smyčky. Původní Langtonova pravidla zůstala téměř všechna zachována. Tímto způsobem se elegantně řeší problém s omezeným prostorem v CA, protože k rozkladu smyčky dojde nejenom při úmrtí smyčky, ale i při nárazu na okraj celulárního prostoru, tj. když se nepodaří dokončit replikaci.

V koloniích SDSR smyček umístněných v omezeném celulárním prostoru můžeme pozorovat pozoruhodné chování. V případě SR smyček dochází k zaplnění celého prostoru statickými vzory složenými z „mrtvol“ smyček, ale u SDSR průběžně umírají a rodí se jedinci a může časem dojít až k „nepředvídatelnému vyhynutí druhu“.



Obrázek 3.17: Ukázka rozkladu SDSR smyčky. (Obrázek převzat z [30].)


Obrázek 3.18: Příklady různých spojených SDSR smyček vzniklých interakcí jejich fenotypů během sebe-replikačního procesu. (Obrázek převzat z [30].)



Obrázek 3.19: Porovnání chování růstu populací SR smyčky a SDSR smyčky v konečném celulárním prostoru. (Obrázek převzat z [19].)

Dalším zajímavým pozorovaným chováním je schopnost spojení dvou nebo více smyček do jedné struktury (příklady na obr. 3.18). Sayama to označil jako *přímou interakci fenotypů* (*direct interaction of phenotypes*). Nicméně toto spojení vede ke ztrátě sebe-replikačních schopností a dostáváme tak „mrtvoly“ v důsledku funkčního selhání. Další informace o smyčce jejím chování naleznete v [30].

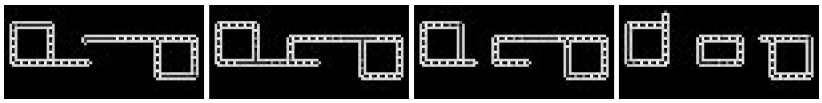
3.3.7 Evoloop smyčka

Evoloop je jméno smyčky, kterou vytvořil Hiroki Sayama a představil v roce 1998 v [31]. Vznikal při studiu chování SDSR smyček a jedná se vlastně o jejich vylepšení. V automatu zanechal 9 stavů a von Neumannovo okolí.

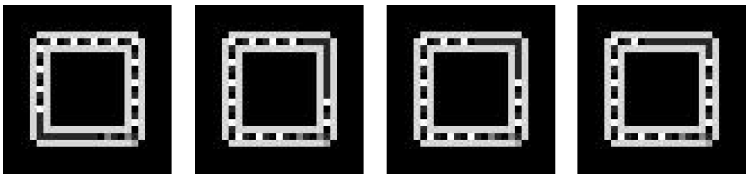
Na konci předchozí kapitoly jsem zmínil, že SDSR smyčky se dokážou spojovat. Toto spíše nechtěné chování v Evoloop je změněno na chtěné a žádané. Struktura smyčky a pravidla chodu automatu jsou stvořeny tak, že při zachování původního principu signálů je při spojení smyček dosaženo jakési evoluce, kdy dochází první ke změně fenotypu a následně k alternaci genotypu smyčky. (Poznámka: Fenotyp zde chápejte jako strukturu nebo tvar smyčky, genotypem jsou myšleny signály proudící ve smyčce.) Mechanismy evoluce nejsou součástí přechodových pravidel CA. Vytvořil několik druhů smyček, které byly strukturálně shodné, ale jejich genotyp byl vylepšen tak, že získaly silnější sebe-replikační schopnosti. Smyčky s rozdílným genotypem představuje obr. 3.21.

Při simulacích Sayama zjistil, že evoluce v jeho automatu spěje ke stále menším smyčkám. Jinak řečeno, čím menší jedinci v populaci, tím rychlejší je jejich schopnost sebe-replikace, což zvyšuje jejich šance na přežití. Uplatňuje se zde princip přírodního (nebo přirozeného) výběru. Příklad lze vidět na obr. 3.22.

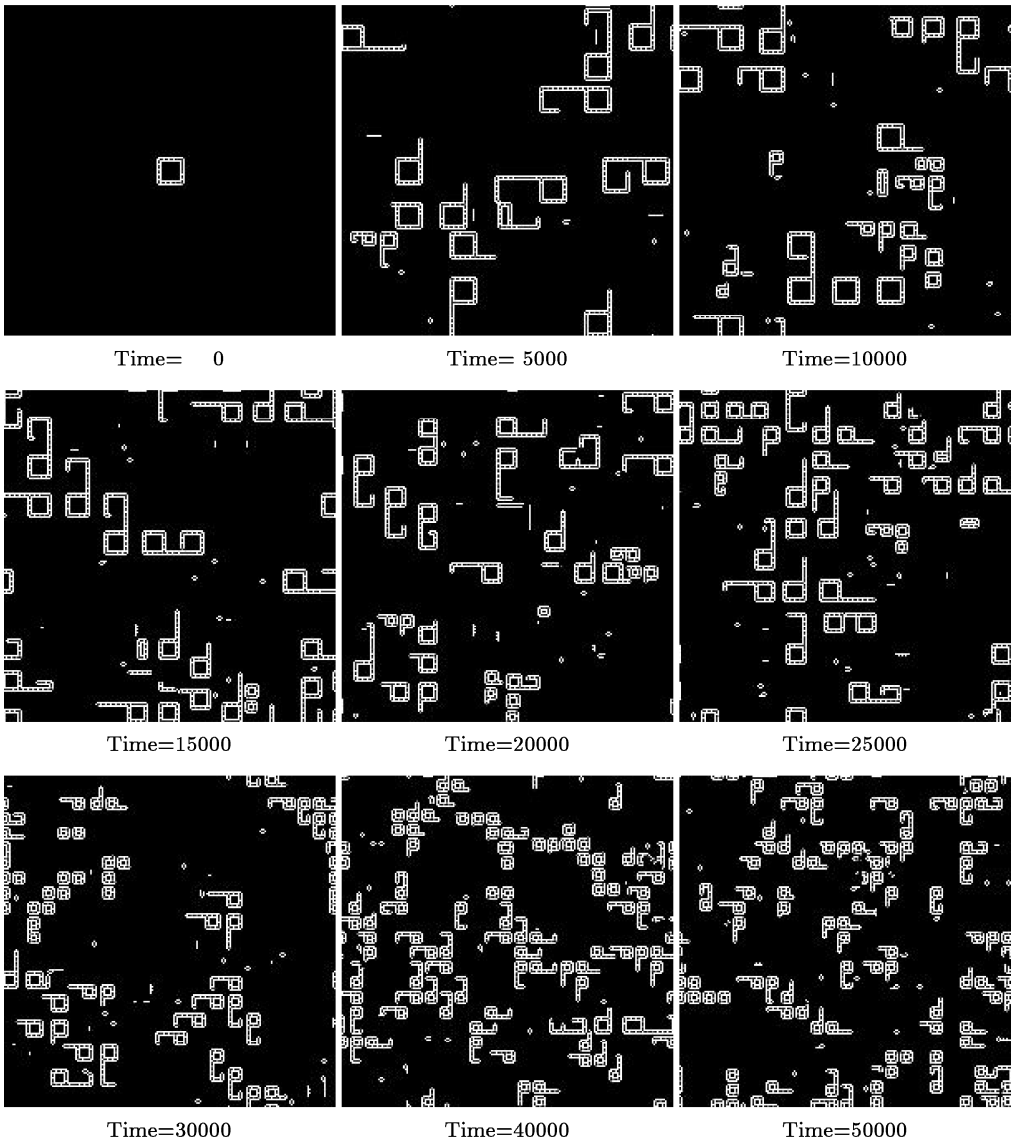
Více informací k tématu naleznete v [31] nebo v [19].



Obrázek 3.20: Převzetí ramena způsobené kolizí dvou Evoloop smyček. (Obrázek převzat z [31].)



Obrázel 3.21: Varianty Evoloop smyček s různou obměnou jejich genomu. (Obrázek převzat z [19].)



Obrázek 3.22: Ukázka vývoje Evoloop smyčky v čase. Celulární prostor má rozměry 200x200 buněk a jsou použity periodické okrajové podmínky. (Obrázek převzat z [19].)

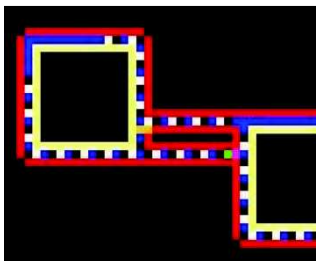
3.3.8 Sexyloop smyčky

V roce pánové Oros a Nehaniv v [32] představili svoji modifikaci Evoloop smyčky nazvanou *Sexyloop*. Vytvořili 2 modely Sexyloop označené jako *M1* a *M2*. Jejich celulární automat byl rozšířen na 10 stavů a pracuje s von Neumannovým okolím. Sexyloop sebe-replikující se smyčky jsou schopny „sexu“ mezi sebou, tj. dokážou si mezi sebou předávat genetický materiál. Vznikly ke studiu možnosti sexu v sebe-reprodukujících automatech a k posouzení dopadu sexu na evoluční proces skrze porovnání Evoloop smyčky a Sexyloop smyček.

Sex je v biologii tedy chápán jako transfer nebo výměna genetického materiálu. Také se dá vysvětlit jako spojení genetického materiálu z více než jednoho zdroje za účelem vytvoření nového jedince. Ve smyčkách se jako genetický materiál chápou proudící signály, takže tyto definice jsou podporou tvrzení, že umělé smyčky, a to konkrétně Sexyloop jsou schopny sexu. Odtud pramení jejich název.

Zavedli pojmy „útočník“ („attacker“) a „napadený“ („attacked“). Útočník je smyčka, která předává svoji genetickou informaci do napadené smyčky. Termín „útočník“ je použit z toho důvodu, že partner předávající dědičnou informaci při sexuálním kontaktu je v evoluční výhodě oproti příjemci, protože příjemce obecně ztrácí při takovémto kontaktu část svého genomu. Toto není obecnou vlastností sexu (jak byl popsán výše), ale omezení současných haploidních sebe-replikujících se modelů smyček, které mají velmi omezené možnosti nést genetický materiál.

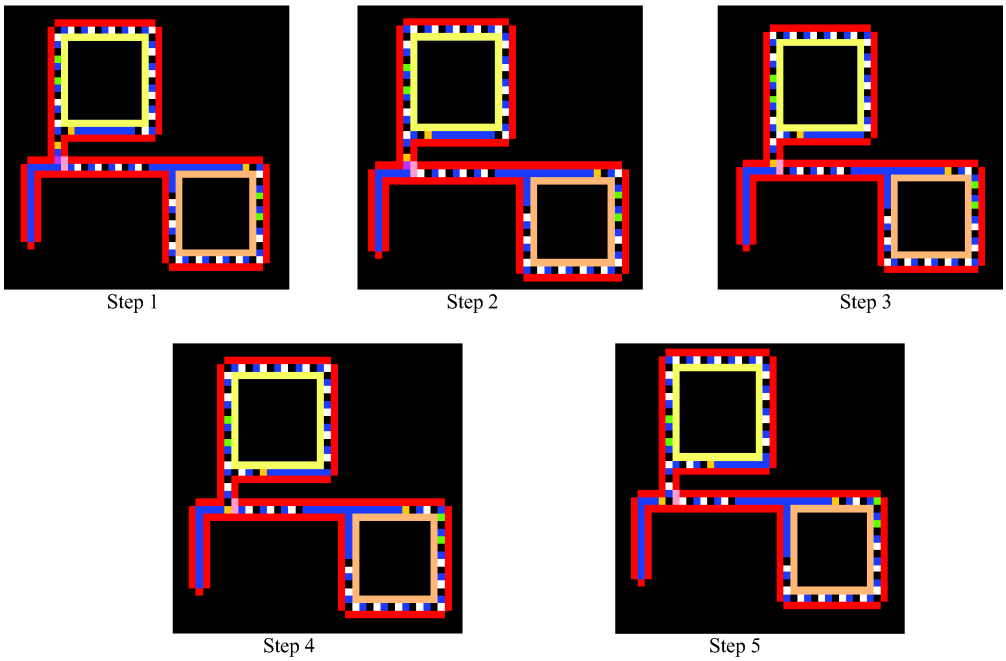
Mezi modely M1 a M2 je rozdíl v mechanismu sexu, v důsledku čehož nejenom že dochází k sexu rozdílným způsobem, ale také se značně liší jejich chování v dlouhodobém evolučním procesu. Oba mechanismy jsou podobné mechanismu rozmnožování baktérií, tzv. bakteriální konjugace [34]. Ukázalo se, že Sexyloop M1 se příliš neliší od Evoloop, zatímco M2 značně změnila evoluční proces změnou genomů individuálních jedinců. Dalším rozdílem je, že Sexyloop smyčky vytváří během procesu menší i větší jedince, oproti Evoloop, kde dochází k tvorbě pouze menších jedinců. Konkrétní principy mechanismu sexu a další informace k Sexyloop naleznete v [32].



Obrázek 3.23: Dvě Sexyloop smyčky mající sex vzájemně mezi sebou. (Obrázek převzat z [32].)

V [33] z roku 2009 je představena další varianta Sexyloop nazvaná *F-sexyloop* (F jako „fertility“, česky plodnost). Tato smyčka je obdařena novými stavy (celkem jich je 12) a pravidly. Hlavní rozdíl je v rozšíření původních Sexyloop smyček o sex gen, který se stará o transfer genetického materiálu mezi smyčkami. Tento gen je analogií k F-faktor plazmidu, který umožňuje bakteriální konjugaci. V závislosti na přítomnosti (F^+) nebo nepřítomnosti (F^-) tohoto faktoru je bakterie schopna jednat jako dárce genetického materiálu. Do umělých smyček pak je tento gen vložen jako jeden stav, který ve formě signálu proudí tělem smyčky. Pokud útočící smyčka neobsahuje tento gen, nedokáže provést transfer svého genetického materiálu do napadené smyčky. Při pokusech se ukázalo, že i když v počáteční populaci bude tento sex gen obsahovat pouze jedna smyčka, tak se postupně během evoluce přenese i do ostatních smyček. Pokud celá populace obsahuje

sex gen, tak pokračujícími kolizemi mezi smyčkami může dojít k vytvoření jedinců, kteří tento gen nemají.



Obrázek 3.24: F-sexyloop smyčky. Ukázka přenosu genetického materiálu z útočící smyčky (horní) do napadené. Sex gen má oranžovou barvu. (Obrázek převzat z [33].)

4 Evoluční návrh pravidel

Bylo třeba zvolit vhodný přístup k nalezení zajímavých pravidel vedoucích k emergentnímu chování. Chtěl jsem se pokusit navrhnout takováto pravidla použitím přístupu automatického návrhu a jako vhodného zástupce jsem zvolil klasický genetický algoritmus.

Tento algoritmus jsem implementoval v jazyce Java. Původně jsem použil k řešení problému knihovnu JGAP (viz. *http://jgap.sourceforge.net/*), ale z důvodu složitosti řešeného problému a značné komplexity této knihovny se mi nepodařilo evoluci pomocí této knihovny korektně zprovoznit. Nakonec jsem provedl vlastní implementaci genetického algoritmu.

4.1 Genetický algoritmus

Evoluční algoritmy patří do skupiny optimalizačních stochastických algoritmů, založených na evoluci populace řešení, přičemž se při prohledávání prostoru řešení využívá fenoménu selekce, křížení a mutace, který byl převzat z evoluce živé přírody [37]. Genetický algoritmus je nejrozšířenějším typem evolučních algoritmů. Jedná se v podstatě o velice účinný mechanismus k prohledávání velkého stavového prostoru.

Činnost genetického algoritmu lze popsat následujícím pseudokódem (převzato z [37]):

1. Nastav t = 0, náhodně generuj počáteční populaci D(0) s mohutností N.
2. Proveď ohodnocení jedinců populace D(t) fitness funkcí F(X).
3. Generuj populaci potomků O(t) s mohutností $M \leq N$ použitím operátorů křížení a mutace.
4. Vytvoř novou populaci D(t+1) nahrazením části populace D(t) jedinci z O(t).
5. Nastav $t \leftarrow t + 1$.
6. Pokud není splněna podmínka pro ukončení algoritmu, jdi na (2).

Blíže se zde teorií genetických algoritmů nebudu zabývat. Více naleznete například v [38].

4.2 Implementace genetického algoritmu

Pro úspěšnou implementaci genetického algoritmu je třeba navrhnout a implementovat následující:

- Kódování problému
- Genetické operátory
  - Mutace
  - Křížení
  - Selekce
- Fitness funkce

Základní rysy tvorby nové populace v mé implementaci genetického algoritmu jsou shrnuty v následujících bodech:

- Polovina nové populace je vytvořena pomocí křížení a mutace selekcí vybraných jedinců.
- Nová generace je doplněna o nejlepšího jedince z předchozí generace (elitismus).

- Zbytek nové populace je doplněno jedinci z rodičovské populace pomocí selekčního operátoru.

4.2.1 Kódování řešení

Pravidlo lze obecně chápat jako zobrazení daného okolí (sousedství) buňky do nového stavu této buňky. Tato pravidla lze vyjádřit různými způsoby. Pro řešení problému sebe-replikace jsem zvolil způsob přímého zápisu pravidel, tj. **fenotypem** je tabulka pravidel tvořená dvojicemi (*okolí buňky*; *nový stav buňky*), kde okolí buňky tvoří vektor stavů buněk z okolí vyšetřované buňky.

Z důvodu zjednodušení implementace jsem zvolil, že program bude napevno pracovat s 5ti buňkovým von Neumannovým sousedstvím. *Okolí buňky* lze potom vyjádřit jako 5ti místné číslo v číselné soustavě o základu udávaného počtem stavů (počet stavů je volitelný parametr; jedná se o velice důležitý parametr, protože při příliš malém počtu stavů evoluce nenajde řešení nikdy a při příliš velkém počtu bude počet všech možných pravidel příliš velký, což povede k velkým paměťovým nárokům a výraznému zpomalení evoluce). Nad okolím tedy existuje uspořádání, kterého využívám v zakódování řešení.

Tabulka pravidel je zakódována ve formě pole (**chromosome**, jinak také označován jako **genotyp**), kde pozice prvku představuje okolí buňky a hodnota na této pozici je *nový stav buňky* (**gen**). V další části textu jsem pro lepší srozumitelnost použil místo pojmu **gen** pojem **pravidlo** a **tabulka pravidel** označuje **chromosom**.

4.2.2 Genetické operátory

Genetický algoritmus pracuje se 3 genetickými operátory: mutace, křížení a selekce.

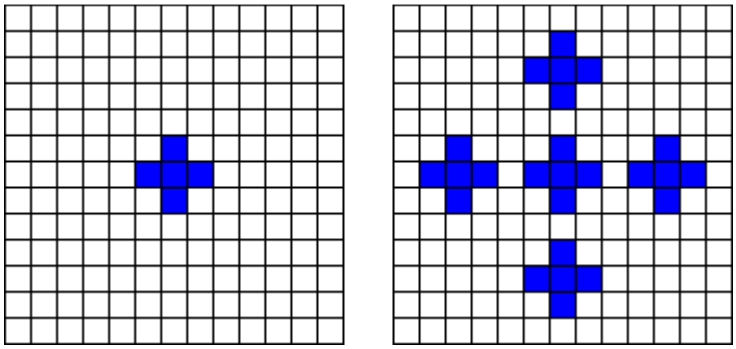
- **Mutace** se provádí se zadanou pravděpodobností. Jelikož by nebylo příliš významné mutovat pouze jedno pravidlo, zejména když může tabulka pravidel obsahovat tisíce, i statisíce pravidel, tak se provádí pro zvolený počet pravidel. Tento počet se generuje náhodně ze zadaného rozsahu minimální míra mutace až maximální míra mutace. Míry se dají zadat jako parametry evoluce. Samotná mutace spočívá v náhodném výběru pravidel, kterým se opět náhodně pozmění jejich hodnota.
- Implementoval jsem 2 druhy **křížení** (jednobodové a uniformní). V obou případech dochází k vytvoření dvou nových jedinců, kterým jsou naplněny tabulky pravidel hodnotami pravidel jejich rodičů. Prochází se simultánně tabulky pravidel obou rodičů a rozhoduje se, které pravidlo se přidá do prvního potomka a které se pravidlo se přidá do druhého potomka. V případě uniformního křížení tato volba probíhá náhodně pro každé pravidlo. U jednobodového křížení se náhodně zvolí bod (pozice v tabulce pravidel, pravidlo), po který se provádí přiřazení pravidla od 1. rodiče do 1. potomka a od 2. rodiče do 2. potomka. Od toho bodu do konce tabulky pravidel se provádí přiřazení pravidla od 1. rodiče do 2. potomka a od 2. rodiče do 1. potomka.
- Jako **selekční operátor** jsem si zvolil a implementoval selekci turnajem. Selekce turnajem pracuje na principu náhodného výběru dvou (případně i více) jedinců z dané populace a nejlepší jedinec (tzn. jedinec s největší hodnotou fitness) z daného výběru je vybrán jako zvolený jedinec.

4.2.3 Výpočet fitness

Výpočet fitness hodnoty jedince je nejnáročnější operací v genetickém algoritmu. Fitness funkce implementuje simulátor celulárních automatů, který provádí vývoj zadané inicializační struktury o zadaný počet kroků, aby následně výsledek porovnal se zadanou cílovou strukturou (nebo strukturami). Výpočet fitness je založen na porovnání odpovídajících si buněk inicializační a cílové struktury, kde počet shodných buněk je hledaná fitness jedince.

4.3 Ověření funkčnosti a zhodnocení

Pro praktické použití bylo nejdříve potřeba implementovaný algoritmus otestovat a zhodnotit možnosti jeho využití při dalším postupu. Jako první test jsem se pokusil pomocí evoluce nalézt pravidla vedoucí k replikaci jednoduché struktury tvaru kříže v celulárním automatu o 3 stavech. Obrázek 4.1 představuje inicializační strukturu a cílovou strukturu, která má být dosažena po 4 vývojových krocích celulárního automatu. Tuto strukturu jsem zvolil záměrně z důvodů její jednoduchosti a relativně malého stavového prostoru k prohledání. Také jsem věděl, že hledané řešení existuje, takže evoluce by na něj měla narazit a skutečně ho v několika stech generacích byla schopna nalézt.



Obrázek 4.1: Inicializační struktura (vlevo) a cílová struktura (vpravo).

Usoudil jsem ale, že takto jednoduchá úloha není dostatečná pro důkladné ověření funkčnosti algoritmu, protože prohledávaný stavový prostor řešení je malý. Jelikož jsem se zaměřil na problematiku sebe-replikujících se smyček, tak se jevilo jako nanejvýš vhodné otestovat genetický algoritmus na nalezení pravidel jedné z nich. Zvolil jsem si Bylovu smyčku, které není příliš komplikovaná a zároveň ani příliš jednoduchá. Abych evoluci maximálně ulehčil práci, tak jsem hledal pravidla chování smyčky krok po kroku jejího vývoje. Inicializační strukturou je originální Bylova smyčka. Cílovými strukturami jsou pak všechny kroky vývoje Bylovy smyčky.

Evoluci jsem rozdělil do několika fází, aby se dala snadno restartovat. Tímto přístupem se mi podařilo nalézt pravidla simulující prvních 21 kroků vývoje Bylovy smyčky. Každé fázi trvalo několik desítek až stovek tisíc generací k nalezení funkčního řešení. Další úspěchy jsem i přes velký počet pokusů s různým nastavením genetického algoritmu neměl. Důvodem tohoto neúspěchu může být malý počet generací, který je způsobený velkou výpočetní náročností problému, v kombinaci s velkým stavovým prostorem. Pro výpočet fitness hodnoty každého jedince v populaci (populaci jsem nastavil na 100 jedinců) je potřeba provést vývoj inicializační struktury a to je výpočetně náročná operace. K nalezení dobrých výsledků je obecně potřeba evoluci spouštět opakovaně na dostatečně dlouhou dobu a s různým nastavením parametrů.

Nechci zde podrobně rozebírat problematiku použití evoluce při návrhu pravidel celulárního automatu, protože to není účelem této práce. Ze získaných zkušeností a výsledků jsem usoudil, že použití evoluce nebude nejvhodnějším přístupem k nalezení pravidel vedoucích k zajímavému chování, protože by bylo potřeba spouštět evoluci opakovaně na velmi dlouhou dobu s nejistým výsledkem. Proto jsem se rozhodl dále se jí již nezabývat a použít raději experimentálního přístupu, s jehož pomocí se mi podařilo nalézt zajímavé výsledky, které jsou představeny v následující kapitole.

5 Urychlení sebe-replikace struktur

První sebe-replikující se struktury byly vlastně velice komplikované systémy schopné univerzální konstrukce, kde sebe-replikace je speciálním případem této konstrukce (viz. kapitoly 3.1 a 3.2). Tyto stroje se staly inspirací ke vzniku výrazně jednodušší struktury schopné pouze sebe-replikace známé jako replikační smyčky (první uvedl Langton, viz. kapitola 3.3.1). Langtonova smyčka odstartovala boom ve vývoji nových sebe-replikujících se struktur. Bylova smyčka (viz. kapitola 3.3.2) přinesla redukci počtu stavů, pravidel a velikosti struktury za pomoci odstranění vnitřní stěny obalu. Chou, Reggia a kol. (viz. kapitola 3.3.3) odstranili obal zcela a dosáhli tak další výrazné redukce pravidel a velikosti struktury. Další smyčky vznikaly s jiným cílem, než je zjednodušení struktury a s tím související redukce pravidel. Tempesti svoji smyčku obdařil schopnostmi konstrukce a výpočtů (viz. kapitola 3.3.4). Podobně Perrierova smyčka (viz. kapitola 3.3.5) má schopnost provádět univerzální výpočet [28]. Hiroki Sayama se vydal zcela jinou cestou a začal se zabývat otázkami života, smrti a evoluce sebe-replikujících se smyček (viz. kapitoly 3.3.6 a 3.3.7). Problém evoluce a sexu mezi smyčkami dále pak rozvedli pánove Oros a Nehaniv (viz. kapitola 3.3.8).

Nové smyčky tedy vznikaly především s cílem redukce počtu stavů, pravidel a velikosti struktury, s cílem rozšíření těchto struktur o nové užitečné schopnosti, nebo za účelem studia zajímavých typů chování. Proč probíhá tak urputný vývoj a výzkum na tomto poli vysvětluje kapitola 3, takže se tím zde nebudu dále zabývat. Chtěl bych se ale zabývat jiným problémem souvisejícím se zlepšováním vlastností sebe-replikujících se smyček, a to je urychlení sebe-replikace. Na toto téma se mi stala velkou inspirací a také velkou motivací práce Tomáše Komendy [36]. V této práci autor docílil urychlení sebe-replikace pomocí nahrazení celulárního automatu vlastním celulárním systémem, který je schopen v určitém vhodném stavu CA provést záměnu sady pravidel za jinou sadu a pokračovat ve vývinu automatu do doby, než se dostane do jiného vhodného stavu CA, který vyústí ve změnu pravidel na původní sadu. Pomocí tohoto přístupu nalezl pravidla vedoucí k urychlení sebe-replikace Bylovy smyčky o 48%. Tato práce ukazuje, že lze nahrazením celulárního automatu jiným vhodným celulárním systémem dosáhnout zajímavých výsledků. Ve své práci jsem se ale rozhodl zůstat u klasických celulárních automatů, protože si nemyslím, že je tato oblast ještě dostatečně prozkoumána a není tedy důvod přecházet k jinému systému. Hledal jsem možnosti, jak dosáhnout urychlení sebe-replikace různých struktur a jaké přístupy lze zvolit k dosažení takovéhoto cíle.

Nalezl jsem pravidla vedoucí k urychlení sebe-replikace Bylovy smyčky a Chou-Reggia smyčky se silnou rotační symetrií (pozn.: Z původní literatury nevyplývá, zda se jedná přímo o smyčku vytvořenou pány Chou, Reggia a spol. Její pravidla jsem nalezl na stránkách H. Sayamy [39] zabývající se sebe-replikačními smyčkami v obecnější rovině. Smyčka má stejnou strukturu jako smyčka na obr. 3.10g, má 6 stavů na buňku a replikuje se v 15 krocích.). Následující kapitoly prezentují výsledky mé práce.

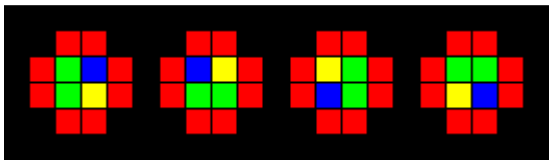
5.1 Upgrade č. 1 Bylovy smyčky

Nejdříve si popíšeme důkladněji originální Bylovu smyčku z pohledu jejího vývoje. Má 6 stavů na buňku, její struktura je tvořena 12 buňkami, replikuje se v 25 krocích, kde tohoto chování je dosaženo pomocí 238 pravidel (viz. příloha 1) pro von Neumannovo okolí. Smyčka se dokáže replikovat do čtyř stran (doprava, nahoru, doleva a dolů). Význam jednotlivých stavů je uveden v tabulce 5.1.

Stav	Význam
0	klidový stav
1	datová cesta
2	vnější obal (stěna)
3	prodlužuje datovou cestu o jeden prvek
4	provede zatočení ramene doleva
5	odpojuje nově vytvořenou smyčku a inicializuje vytváření nové kopie

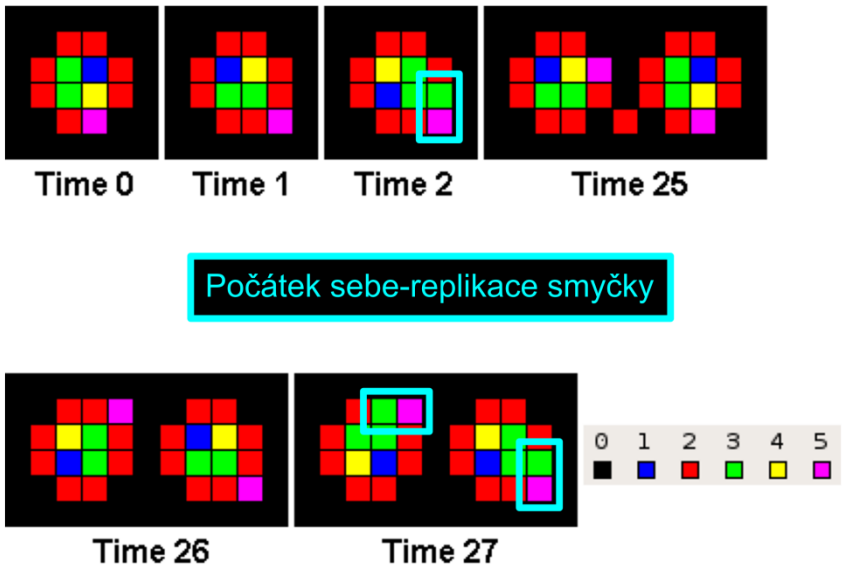
Tabulka 5.1: Význam jednotlivých stavů Bylovy smyčky.

Stavy 3 a 4 neustále rotují uvnitř obalu smyčky proti směru hodinových ručiček a tvoří spolu se stavem 1 jádro smyčky o 4 buňkách. Samotné jádro uzavřené v obalu nezpůsobuje žádnou činnost smyčky, vyjma svojí vlastní rotace.



Obrázek 5.1: Ukázka všech 4 možných natočení jádra Bylovy smyčky.

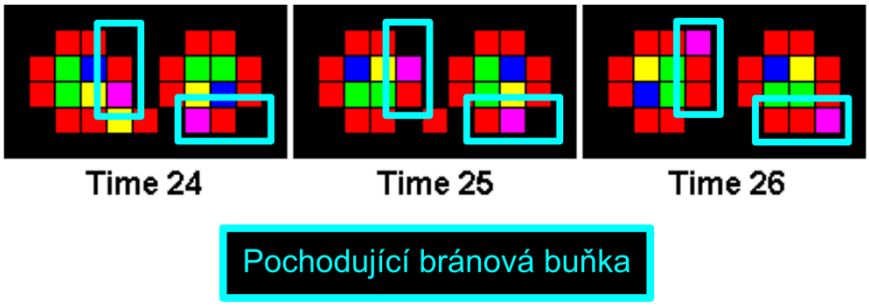
Pro započetí replikace smyčky je proto třeba zaměnit jednu z buněk tvořících obal (stav 2) buňkou o stavu 5, která inicializuje vytváření kopie smyčky. Takto dostáváme již naši starou známou Bylovu smyčku (viz. obr. 5.2, time 0). Stav 5 cestuje po směru rotace jádra do rohu smyčky (obr. 5.2, time 1 a 26), kde plní úlohu jakési brány, která supluje úlohu ramene známého z Langtonovy smyčky. V následném kroku smyčka započne svoji replikaci. Na obrázku 5.2 je také vidět, že po 25 krocích je vytvořena přesná kopie původní smyčky, která započne replikaci do stejné strany, do jaké se replikovala její rodičovská smyčka. Ta se natočí o 90° proti směru hodinových ručiček a zahájí replikaci tímto novým směrem.



Obrázek 5.2: Ukázka zahájení sebe-replikace Bylovy smyčky.

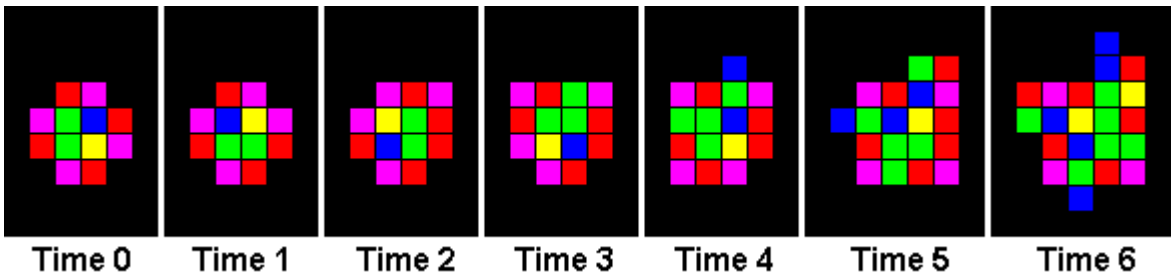
Stav 5, který mimo jiné plní úlohu brány, vykazuje jistou podobnost s Tempestiho smyčkou (viz. kapitola 3.3.4). Ta se na rozdíl od Bylovy smyčky dokáže replikovat do všech 4 možných směrů naráz. To se stalo i mojí inspirací a obohatil jsem Bylovu smyčku o schopnost vícenásobné sebe-replikace.

Zde hraje důležitou úlohu právě stav 5. Myšlenka je přidat podobně jako Tempesti na každou stranu smyčky jednu bránovou buňku. Přidání 3 dalších brán na odpovídající místa do původní Bylovy smyčky však vede k téměř jejímu okamžitému selhání, protože pro správný průběh replikace je potřeba, aby bylo také jádro smyčky natočeno ve správné poloze. Chtěl jsem se vyhnout výraznému zásahu do původní smyčky a jejích pravidel, proto jsem její chování analyzoval a pro její vylepšení jsem využil „pochodování“ bránové buňky při ukončování replikace smyčky. Obrázek 5.3 ukazuje, jak v původní a v nově vzniklé smyčce „pochoduje“ brána po obalu smyčky až do jejího rohu. Je vidět, že bránová buňka se pohybuje současně se stavem 4.



Obrázek: 5.3: Ukázka „pochodování“ bránového stavu v originální Bylově smyčce.

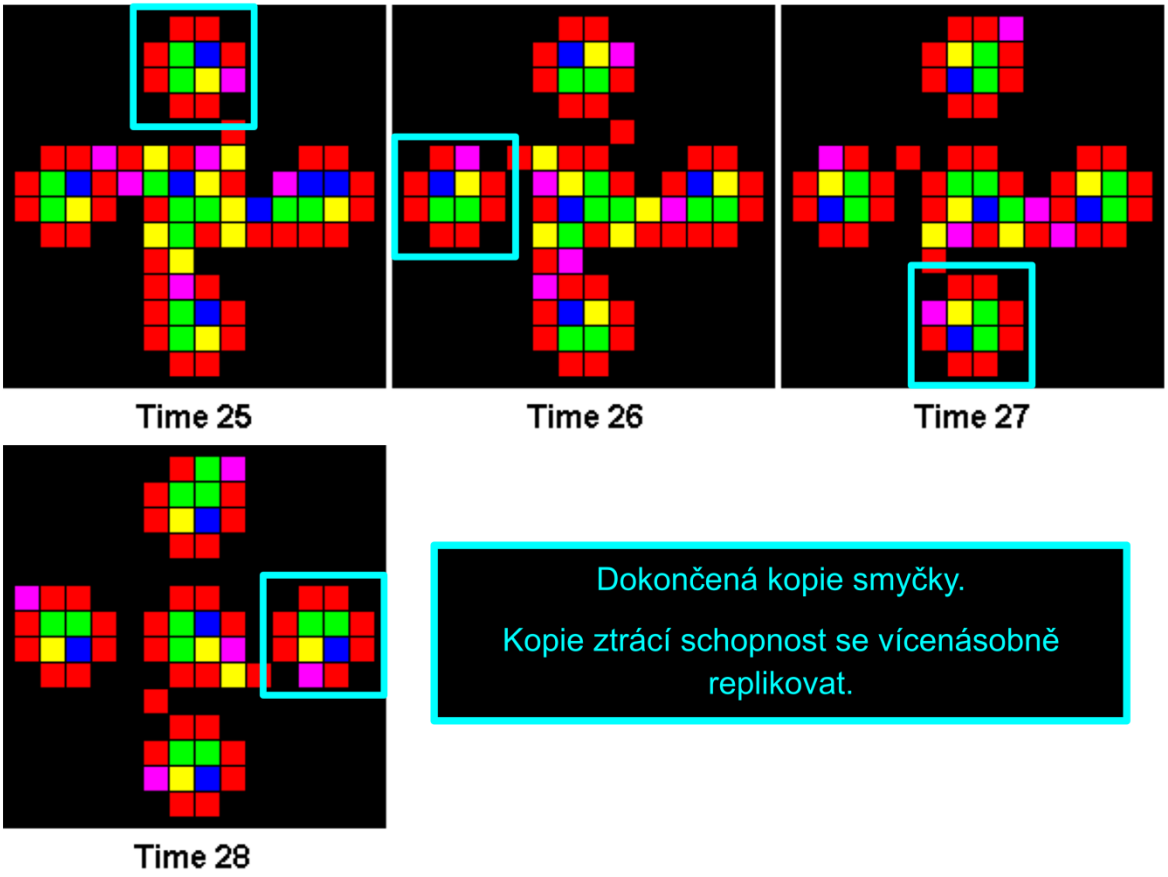
Rozšířením původní sady pravidel o 14 nových pravidel (celkem tedy smyčka používá 252 pravidel ze 7 776 možných; viz. příloha 2) a pozměněním počáteční struktury Bylovy smyčky (obr. 5.4, time 0) jsem dosáhl požadovaného chování. Počáteční smyčka se replikuje do 4 stran naráz. Nejedná se o zcela simultánní vícenásobnou sebe-replikaci, protože k zahájení sebe-replikace je potřeba, aby se natočilo jádro do správné pozice, tzn. že další kopie se začne vytvářet se zpožděním jednoho kroku. Proces replikace jedné smyčky zabere 25 kroků. Za 28 kroků jsou vytvořeny 4 repliky počáteční smyčky. Takže urychlení nespočívá v redukci potřebného počtu kroků, ale ve vícenásobné replikaci. Následující obrázek ukazuje zahájení replikace ve všech 4 směrech. V kroku 3 se začíná replikovat smyčka směrem nahoru, v kroku 4 směrem vlevo, v kroku 5 směrem dolů a v kroku 6 směrem doprava.



Obrázek 5.4: Vylepšená Bylova smyčka + prvních 6 kroků jejího vývoje.

Výraznou vlastností tohoto vylepšení je, že vícenásobná sebe-replikace probíhá pouze pro počáteční smyčku. Všechny její repliky totiž ztrácejí 3 ze 4 bránových buněk, takže dále už probíhá sebe-replikace podle stejného scénáře jako u originální Bylovy smyčky. Tuto vlastnost ilustruje

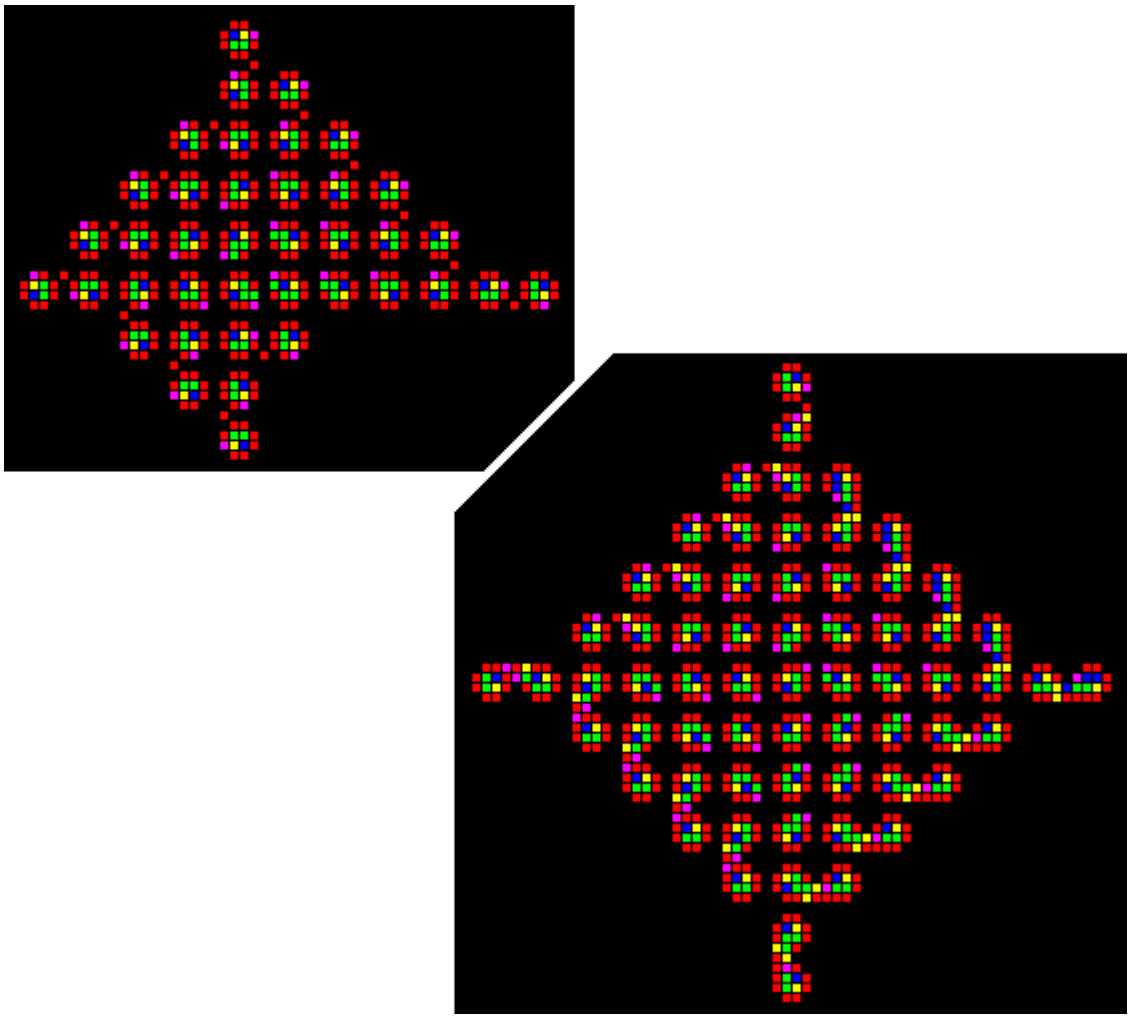
obrázek 5.5. Vyznačené smyčky jsou dokončené kopie původní smyčky (obr. 5.4, time 0), ale místo původních 4 bran mají pouze bránu jednu.



Obrázek 5.5: Dokončení replikace prvních 4 kopií smyčky.

Urychlení u tohoto vylepšení spočívá v počáteční vícenásobné sebe-replikaci. V dalších fázích oproti původní smyčce již nemá žádné další vylepšení. Na obrázku 5.6 se nacházejí 2 kolonie smyček. Obě kolonie vznikly z jedné počáteční smyčky po 150ti krocích vývoje, jedna pro původní Bylovu smyčku, druhá pro vylepšenou verzi. Na první pohled je vidět rozdíl ve strukturách kolonií. Kolonie vlevo, patřící původní smyčce, se vyznačuje spirálovitým vzorem růstu, zatímco vylepšení smyčky přineslo i zlepšení ve struktuře růstu kolonie. Struktura je podobně jako u Tempestiho smyčky pravidelná.

Pokud započítáme i smyčky, které ještě nebyly zcela dokončeny, tak po 150ti krocích přineslo představené vylepšení 40% nárůst počtu smyček. S přibývajícím počtem generací smyček toto procento klesá, ale to není tady příliš podstatné. Vylepšení Bylovy smyčky upgrade č. 1 je pouze mezikrokem k daleko výraznějšímu urychlení, které představuje následující kapitola.



Obrázek 5.6: Kolonie smyček po 150ti krocích. Vlevo pro původní Bylovu smyčku. Vpravo pro vylepšenou smyčku.

5.2 Upgrade č. 2 Bylovy smyčky

Možnost vícenásobné sebe-replikace představená v předchozí kapitole je jistě důležitá, ale mě především zajímá urychlení sebe-replikace ve smyslu zmenšení potřebného počtu kroků k získání kopie replikované smyčky. Protože jsem chtěl zůstat u klasických celulárních automatů, tak jsem měl pouze 2 možné cesty, jak dosáhnout urychlení - pokusit se zcela předělat pravidla Bylovy smyčky nebo rozšířit stavový prostor o 1 nebo více stavů a najít nová pravidla vedoucí k dosažení požadovaného cíle. 2. varianta mi po zralé úvaze přišla jako schůdnější.

Rozšířil jsem počet stavů z 6 na 7. Zachoval jsem strukturu smyčky představenou v předchozí kapitole (viz. obr. 5.4, time 0). Provedl další vylepšení pravidel Bylovy smyčky, které přineslo urychlení sebe-replikace smyčky z 25ti kroků na 17 kroků. Dále jsem více rozvedl problematiku vícenásobné sebe-replikace tak, že noví jedinci neztrácí zcela tuto schopnost.

Po přidání dalšího stavu je potřeba identifikovat v procesu replikace místo, kde bude vhodné tento nový stav vložit. Podobně, jako ve své práci T. Komenda, i já jsem usoudil, že nejvhodnější bude 10. krok, kde se osamostatňuje stav 1 (viz. obr. 5.7), který se v následném kroku změní na nový stav 6.

Obrázek 5.7: 10 krok replikace Bylovy smyčky – klíčový prvek pro urychlení sebe-replikace.

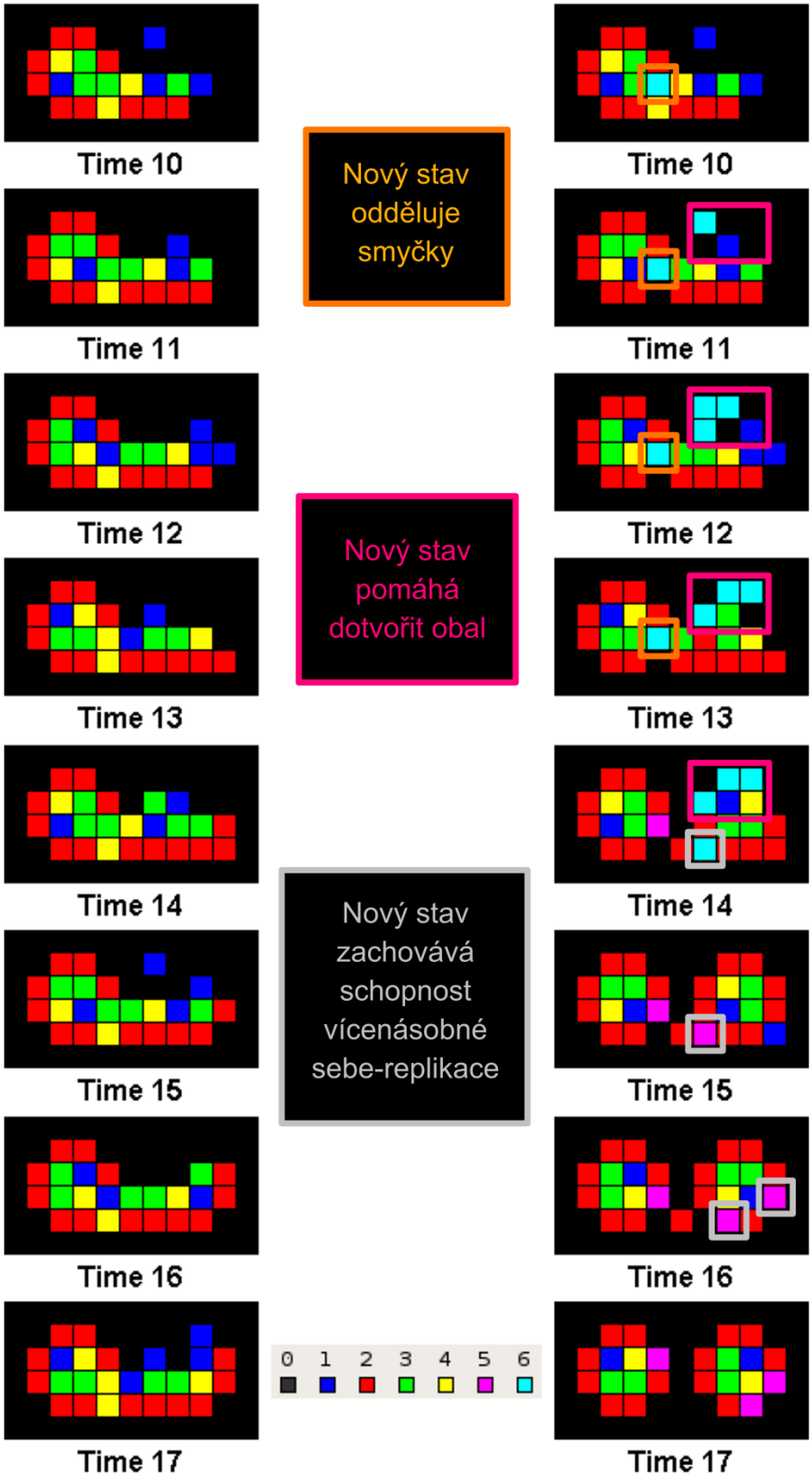
Základní význam jednotlivých stavů z původní Bylovy smyčky zůstává nezměněn. Nový stav 6 má 3 hlavní úkoly:

1. Odděluje nově vznikající smyčku od rodičovské.
2. Pomáhá rychleji dotvořit obal nově vznikající smyčky.
3. Umožňuje zachovat novému jedinci schopnost vícenásobné sebe-replikace.

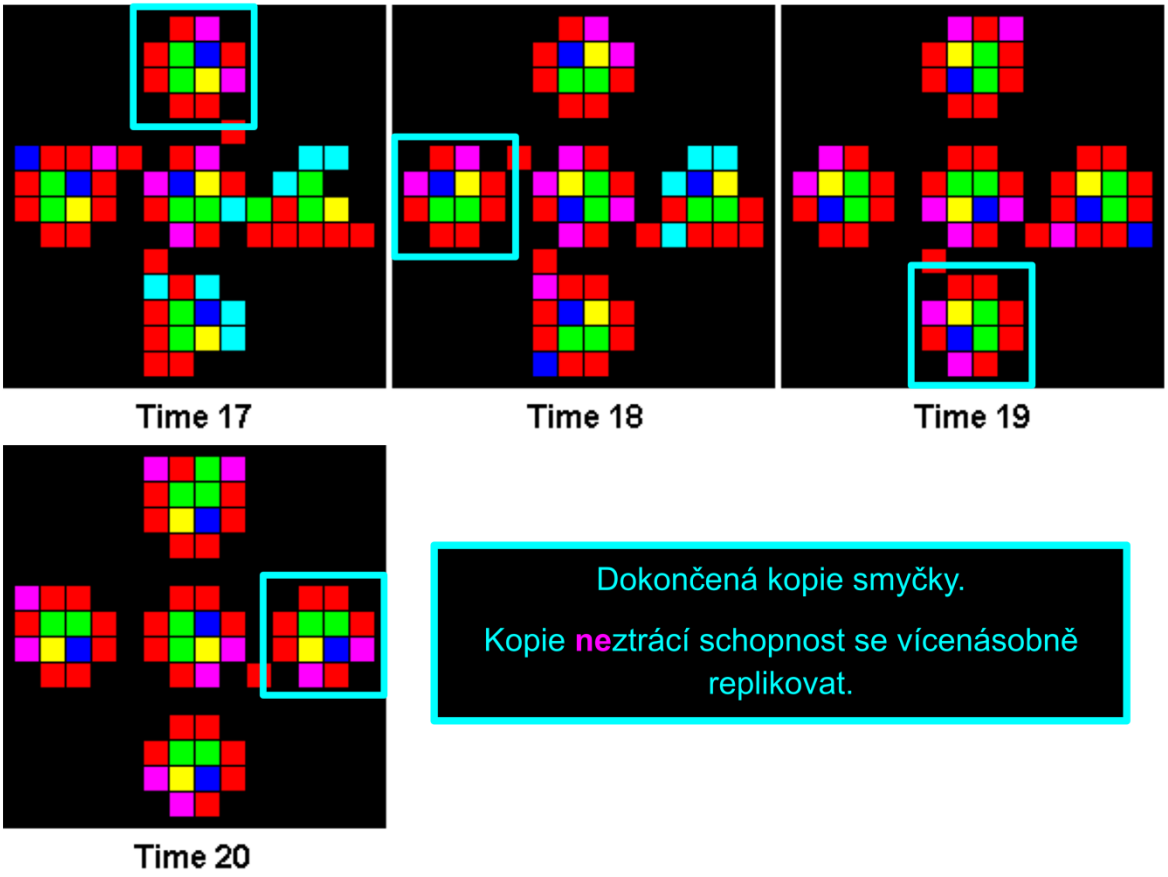
Obrázek 5.8 názorně představuje popsanou funkčnost tohoto nového stavu. Obrázek zároveň provádí srovnání s vývojem smyčky podle původních pravidel pro 10. až 17. krok vývoje smyčky. Jelikož mnou vylepšená Bylova smyčka má i upravenou počáteční strukturu, takže by srovnání bylo příliš nepřehledné a nenázorné, tak jsem pro tuto ukázku pozměnil strukturu smyčky na tu původní, představenou J. Bylem. V 10. kroku se objevuje první buňka se stavem 6, jejímž úkolem je separovat replikovanou smyčku od nově se formujícího jedince. Toto je velice důležitý okamžik, který zabraňuje rodičovské smyčce posílat další signály způsobující sebe-replikaci. Takto je umožněno poslat pouze 2 kompletní sekvence signálů 3 - 3 - 4, oproti originální smyčce, která posílá až 6 těchto sekvencí. V 11. kroku se mění osamocený stav 1 na stav 6, který v dalších 4 krocích pomůže dotvořit obal smyčky.

Na obrázku 5.8 je také vidět, že i když smyčka na počátku má pouze jednu bránovou buňku, tak její potomek má již tyto buňky dvě. Jedná se o vlastnost, kterou jsem napevno zakódoval do pravidel. Smyčka může mít na počátku 1, 2, 3 nebo i 4 brány, ale její potomci budou mít vždy zásadně 2 brány. Tuto vlastnost ilustruje také Obrázek 5.9, kde replikovaná smyčka se 4mi bránami má po 20. vývojovém kroku 4 potomky s 2mi bránami. Důvody, proč jsou to jen 2 brány a ne rovnou 4 jsou dva. Prvně je velice obtížné nalézt pravidla, která by vedla k takovému chování a druhý a podstatnější důvod je, že je to zcela zbytečné. I při pouze dvojnásobné sebe-replikaci jsou zabrány všechny možné prostory, kam se může smyčka replikovat a pak není nutné řešit problém s kolizí dvou vznikajících smyček, která by vedla k havárii smyček. Řešení kolizí je velice náročný proces.

Popsaného chování jsem dosáhl rozšířením původní sady pravidel o 120 nových pravidel, 68 pravidel jsem ubral a 9 pozměnil. Celkem se používá 290 pravidel z 16 807 možných (viz. příloha 3).

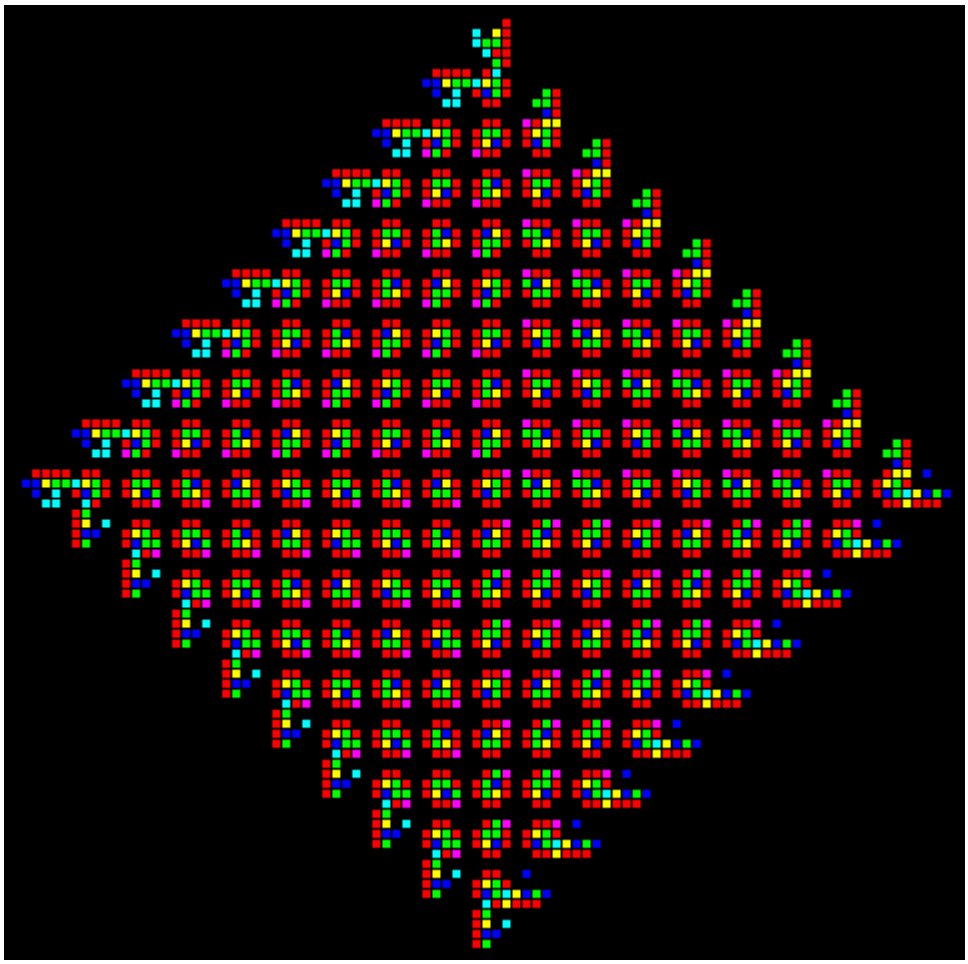


Obrázek 5.8: Srovnání 10. až 17. kroku vývoje původní struktury Bylovy smyčky pro originální pravidla Bylovy smyčky (vlevo) a pro mnou urychlenou verzi (vpravo).

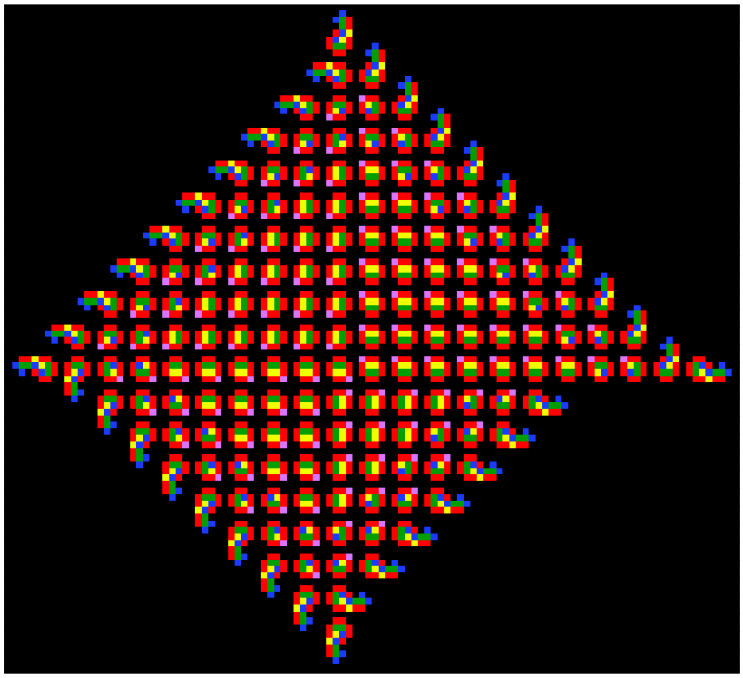


Obrázek 5.9: Dokončená replikace prvních 4 kopií smyčky.

Urychlení zde tedy spočívá ve dvou použitých přístupech. Je zredukovaný počet potřebných kroků k vytvoření repliky smyčky a dále jsou smyčky obdařeny schopností vícenásobné sebe-replikace, díky čemuž je struktura růstu kolonie smyček pravidelného tvaru (viz. obr. 5.10). T. Komenda udává, že jeho vylepšení urychluje sebe-replikaci smyčky o 48%. Moje práce urychluje smyčku pouze o 32% (nepočítám vícenásobnou sebe-replikaci). Celkové urychlení oproti originální Bylově smyčce se dá jen těžko určit kvůli rozdílné rychlosti růstu smyček. Provedu pouze prosté srovnání po 150. kroku simulace. Kolonie smyček pracujících s pravidly upgrade č. 2 po 150ti krocích obsahuje 145 kompletních smyček (viz. obr. 5.10). Oproti tomu u originální Bylovy smyčky je to pouze 39 smyček, a to znamená nárůst o více jak 73%, což ve výsledku není vůbec málo. Pro srovnání na obrázku 5.11 si můžete prohlédnout kolonii smyček po 150ti krocích z práce T. Komendy. Zde se nebudu pouštět do srovnávání v procentuálním nárůstu množství smyček, protože u jeho smyček dochází k úmrtí jedinců, kteří nemají již prostor ke své replikaci (více informaci o tom naleznete v jeho práci [36]).



Obrázek 5.10: Kolonie urychlených smyček po 150ti krocích.



Obrázek 5.11: Kolonie smyček urychlených T. Komendou po 150ti krocích. (obrázek převzat z [36])

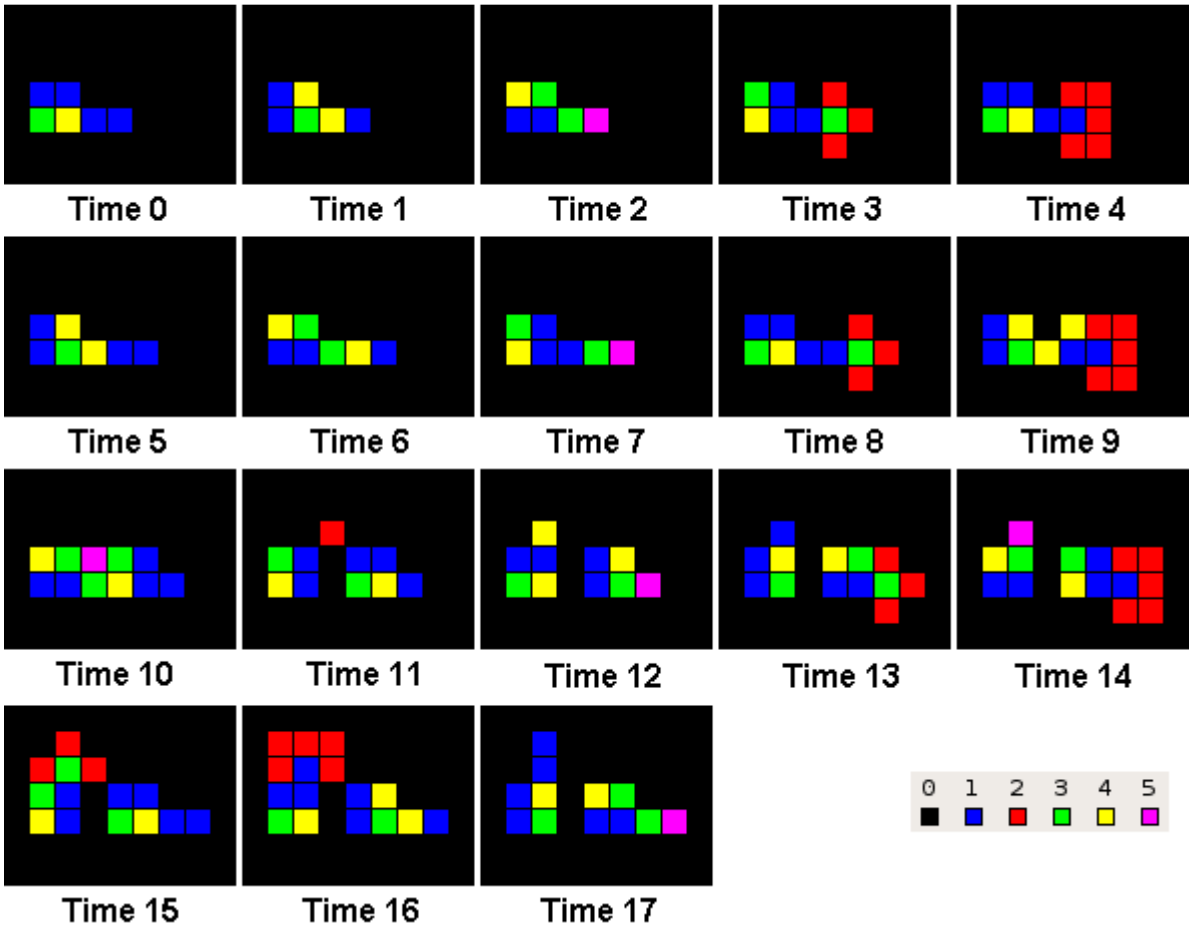
5.3 Upgrade č. 1 Chou-Reggia smyčky

Nejdříve popíšu důkladněji původní Chou-Reggia smyčku z pohledu jejího vývoje. Má 6 stavů na buňku, strukturu tvořenou 6 buňkami, replikuje se v 15 krocích, kde tohoto chování je dosaženo pomocí 137 pravidel (viz. příloha 4) pro von Neumannovo okolí. Smyčka se dokáže replikovat do 4 stran (doprava, nahoru, doleva a dolů). Vypozorovaný přibližný význam jednotlivých stavů je uveden v tabulce 5.2.

Stav	Význam
0	klidový stav
1	datová cesta
2	podílí se na prodlužování ramene
3	separuje smyčky + další pomocné funkce
4	inicializuje prodlužování ramene
5	podílí se na prodlužování ramene

Tabulka 5.2: Přibližný význam jednotlivých stavů Chou-Reggia smyčky.

Strukturu smyčky a prvních 17 vývojových kroků představuje obr. 5.12. Smyčka je oproti Bylově vybavena ramenem. Po 15ti krocích se vytvoří replika, ale ještě další 2 kroky jsou potřeba, aby se u rodičovské smyčky vytvořilo rameno v nové pozici (smyčka se natočí o 90° proti směru hodinových ručiček).

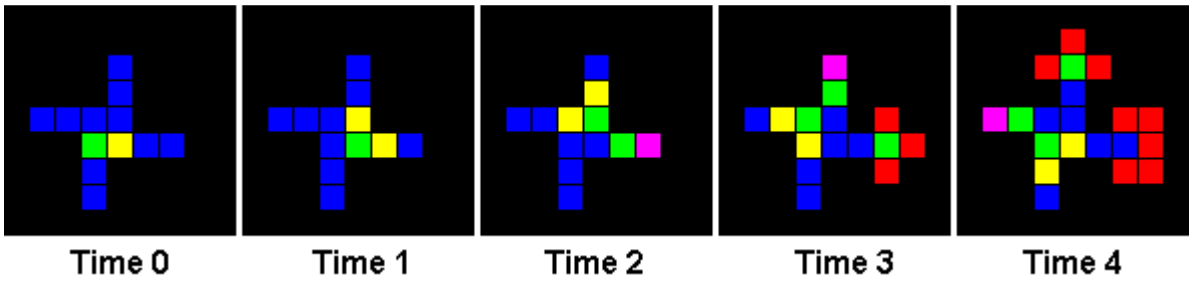


Obrázek 5.12: Struktura Chou-Reggia smyčky + prvních 17 vývojových kroků.

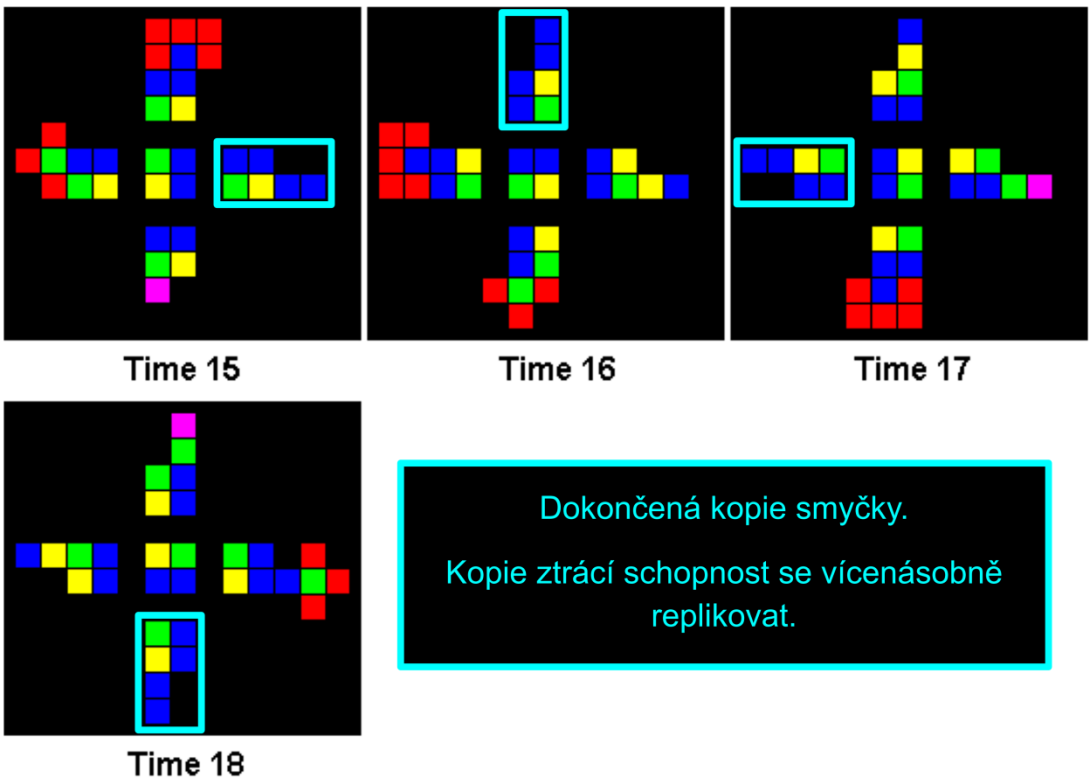
Podobně jako v Bylově smyčce i zde je základ rotující jádro. Pokud ubereme smyčce rameno, tak jádro bude neustále rotovat bez další jiné činnosti. Jádro je tvořeno stavy 3 a 4 spolu se stavem 1.

Urychlení sebe-replikace smyčky jsem pojal stejně jako v kapitole 5.1, tj. rozšířil jsem Chou-Reggia smyčku o schopnost vícenásobné sebe-replikace. V první fázi bylo třeba upravit strukturu smyčky. Tady je to velice snadná úloha - stačí přidat 3 nová ramena, jak ukazuje obr. 5.14, time 0. Dalším krokem je úprava pravidel tak, aby se smyčka replikovala do nově přidaných směrů a nedošlo k havárii smyčky. Opět se nejedná o zcela simultánní vícenásobnou sebe-replikaci, protože k zahájení replikace v daném směru je třeba správné natočení jádra smyčky. Ilustruje to následující obrázek. V kroku 1 se zahajuje replikace směrem doprava, v kroku 2 směrem nahoru, v kroku 3 směrem nalevo a v kroku 4 směrem dolů.

Obrázek 5.13: Ukázka všech 4 možných natočení jádra Chou-Reggia smyčky.



Obrázek 5.14: Vylepšená struktura smyčky + prvních 4 kroky jejího vývoje.

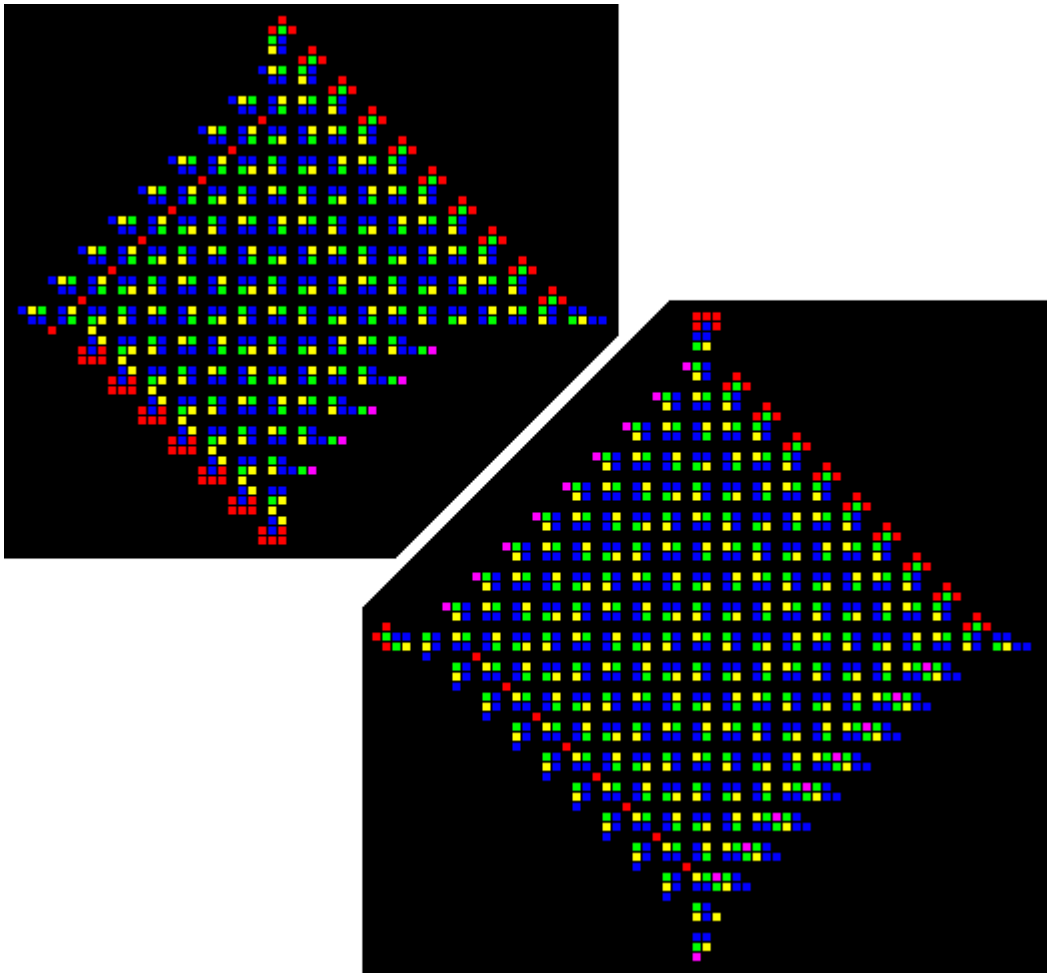


Obrázek 5.15: Dokončení replikace prvních 4 kopií smyčky.

Tohoto chování jsem dosáhl rozšířením sady pravidel o 7 nových pravidel. Celkem tedy smyčka používá 144 pravidel ze 7 776 možných (viz. příloha 4).

Stejně jako u vylepšení představeného v kap. 5.1 je výraznou vlastností tohoto řešení, že nově vytvořené kopie smyček ztrácí schopnost vícenásobné sebe-replikace. Obrázek 5.15 toto chování ilustruje. Vyznačené smyčky jsou dokončené kopie původní smyčky, ale místo původních 4 ramen disponují pouze ramenem jedním.

Toto vylepšení tedy kromě počáteční vícenásobné sebe-replikace nepřináší žádné další urychlení. Opět ale přináší změnu ve struktuře růstu kolonie smyček. Oproti původní Chou-Reggia smyčce, kde je vzor spirálovitý, tak moje vylepšení přináší vzor pravidelný (viz. obr. 5.16).



Obrázek 5.16: Kolonie smyček po 150ti krocích. Vlevo pro původní Chou-Reggia smyčku. Vpravo pro vylepšenou smyčku.

5.4 Upgrade č. 2 Chou-Reggia smyčky

V této kapitole představím urychlení Chou-Reggia smyčky spočívající v umenšení počtu potřebných kroků k vytvoření repliky smyčky. Tohoto jsem dosáhl rozšířením počtu stavů z 6 na 7. Zachoval jsem strukturu představenou v předchozí kapitole (viz. obr. 5.14, time 0). Provedl další vylepšení pravidel Chou-Reggia smyčky, které přineslo urychlení z 15 potřebných kroků na 11 kroků.

Opět podobně jako v kapitole 5.2 je potřeba identifikovat v procesu replikace místo, kde se nový stav objeví. Jako nejvhodnější místo jsem objevil ve 4. vývojovém kroku dvojici rohových buněk (viz. obr. 5.17), které se v následujícím kroku mění na nový stav.

Obrázek 5.17: 4. Krok replikace Chou-Reggia smyčky – místo pro zavedení nového stavu.

Význam jednotlivých stavů z původní Chou-Reggia smyčky zůstává přibližně totožný. Nový stav 6 má tyto hlavní úkoly:

1. Odděluje nově vznikající smyčky.
2. Pomáhá rychleji dotvořit tělo nově vznikající smyčky.
3. Pomáhá rodičovské smyčce zachovat schopnost replikace.

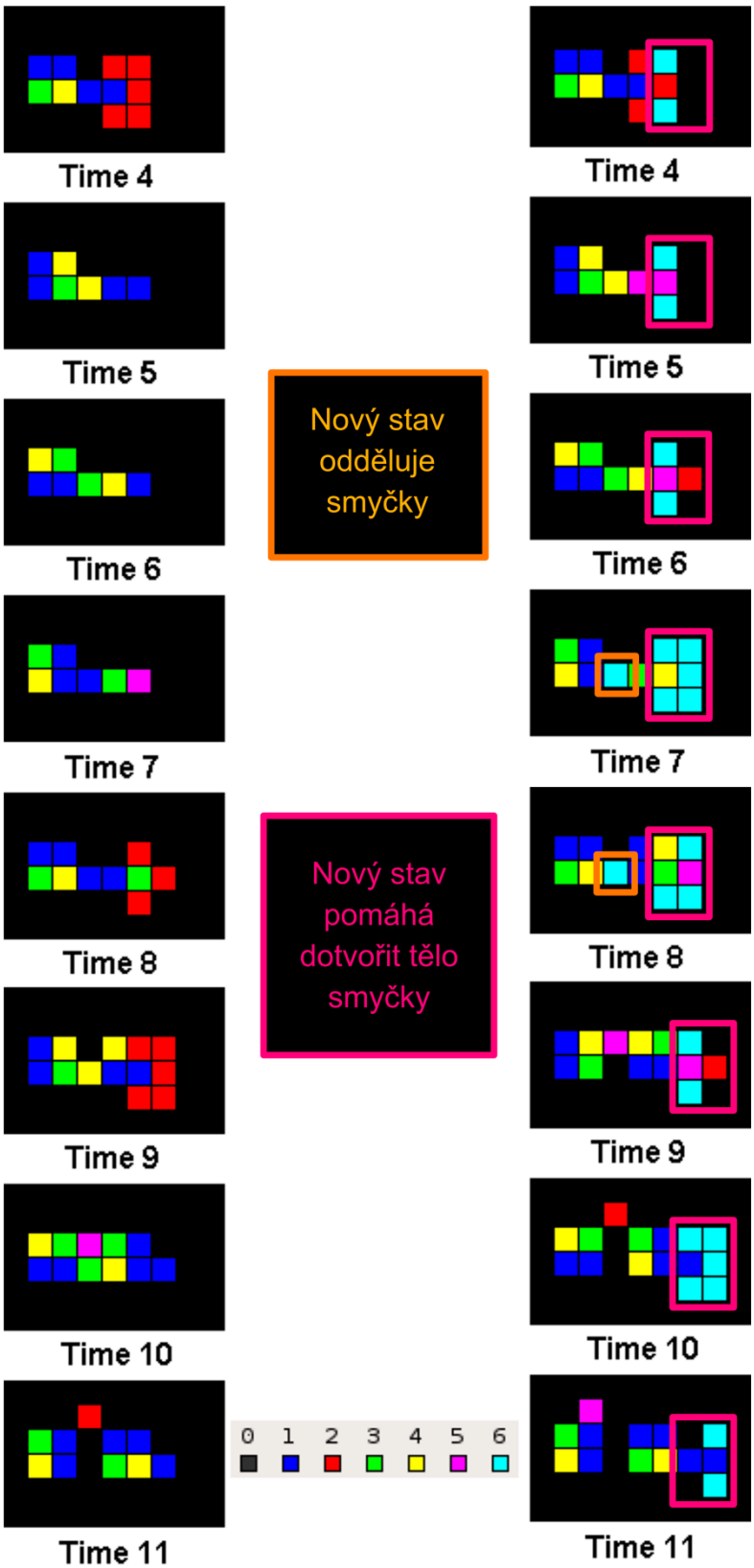
Na obrázcích 5.18 a 5.19 si můžete názorně prohlédnout popsanou funkčnost nového stavu. Obrázek 5.18 zároveň provádí srovnání s vývojem smyčky podle původních pravidel pro 4. až 11. krok vývoje smyčky. A jelikož má vylepšená smyčka i upravenou počáteční strukturu, tak jsem prováděl srovnání pro původní strukturu Chou-Reggia smyčky. V kroku 4 se objevují první 2 buňky o stavu 6. Tyto buňky pomáhají rychleji dotvořit tělo nově vznikající smyčky, tj. jádro smyčky + rameno, aby smyčka byla schopná další sebe-replikace. Jedná se zde o značně složitější proces, než byl u Bylovy smyčky, protože smyčka postrádá obal, který by usměrňoval chování buněk. Je třeba rozumně navrhovat pravidla, aby nedošlo k nekontrolovanému růstu počtu aktivních buněk (tj. buňky, které nejsou v klidovém stavu).

V kroku 7 se objevuje buňka se stavem 6 sloužící k separaci replikované smyčky od nově formujícího se jedince. Zamezuje tím posílání dalších signálů z rodičovské smyčky způsobujících prodlužování ramene. Takto je pouze 2x zaslána sekvence signálů 4 - 3, oproti originální smyčce, která posílá 3 tyto sekvence.

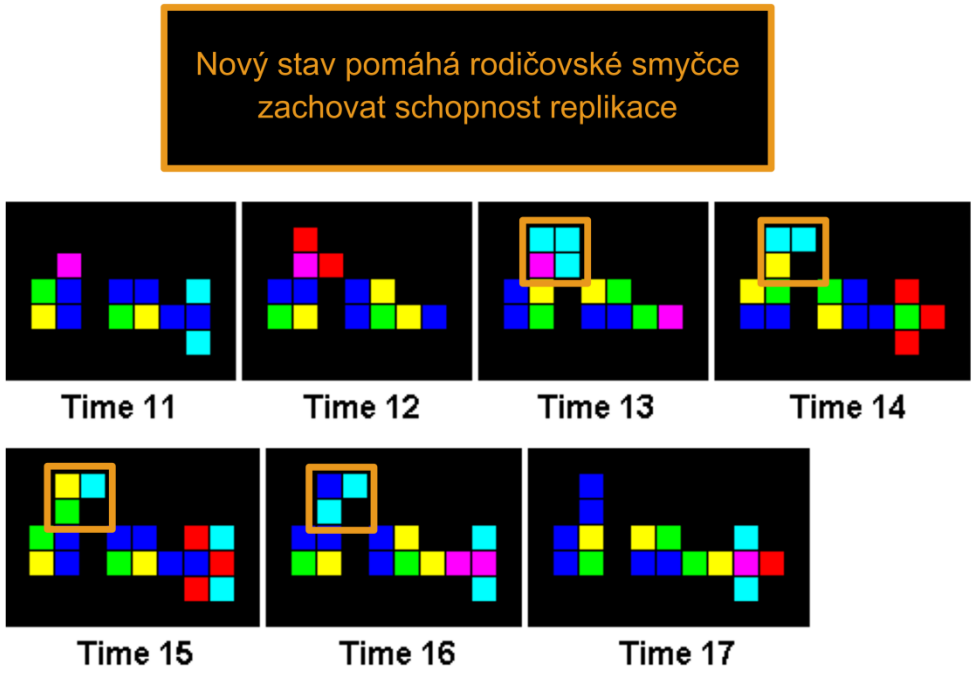
Jako velice komplikované se ukázalo zachování schopnosti sebe-replikace u rodičovské buňky. V originální smyčce (jak je vidět na obr. 5.12, time 15, 16 a 17) dochází k dokončení ramene rodičovské buňky během dvou kroků od dokončení potomka. U mojí vylepšené verze je těchto kroků potřeba 6 (viz. obr. 5.19). Tato nepříjemnost je způsobená špatně natočeným jádrem rodičovské smyčky. Pomocí nového stavu se mi ale podařilo rameno vystavět. Shodou okolností se jedná o přesně stejný vývojový krok, kdy dochází k dokončení nového ramene rodičovské smyčky jako u originální smyčky.

Bohužel se mi nepodařilo nalézt taková pravidla, která by obdařila repliky smyček schopností vícenásobné sebe-replikace. Jak už jsem uvedl výše, smyčka postrádá obal, což velice ztěžuje veškeré pokusy o vytvoření dalších ramen u potomků.

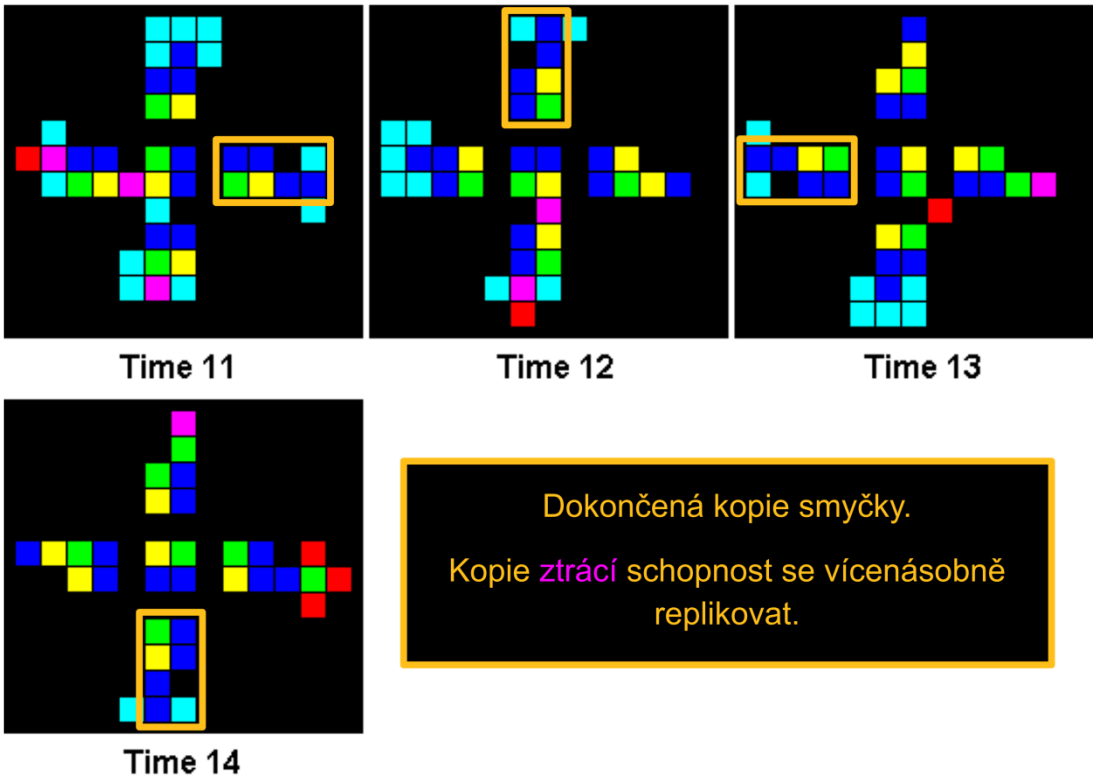
Popsaného chování jsem dosáhl rozšířením původní sady pravidel o 156 nových pravidel, 76 pravidel jsem ubral a 8 pozměnil. Celkem se používá 217 pravidel z 16 807 možných (viz. příloha 6).



Obrázek 5.18: Srovnání 4. až 11. kroku vývoje původní struktury Chou-Reggia smyčky pro originální pravidla Chou-Reggia smyčky (vlevo) a pro urychlenou verzi.



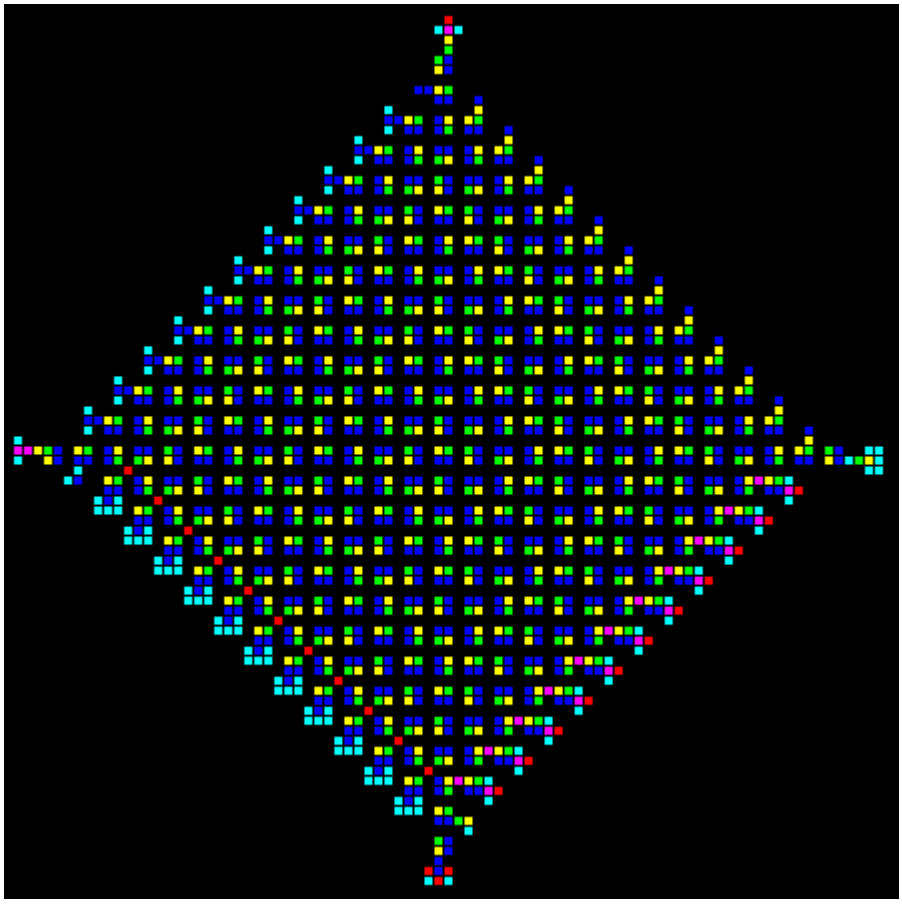
Obrázek 5.19: 11. až 17. krok vývoje urychlené verze smyčky pro původní strukturu Chou-Reggia smyčky.



Obrázek 5.20: Dokončená replikace prvních 4 kopií u vylepšené verze smyčky.

Oproti urychlení Bylovy smyčky představeného v kapitole 5.2 ztrácí nový jedinec schopnost vícenásobné sebe-replikace, která by jinak přinesla výrazný nárůst počtu replik smyček. Umenšení počtu potřebných kroků k sebe-replikaci z 15 na 11 kroků představuje urychlení o cca 26,67% (bez

vícenásobné replikace na počátku). A podobně jako v kapitole 5.2, tak i zde provedu pouze prosté srovnání po 150. kroku simulace z důvodu rozdílné rychlosti růstu smyček. Kolonie smyček pracujících s pravidly upgrade č. 2 po 150ti krocích obsahuje 295 kompletních smyček (viz. obr. 5.21). Oproti tomu u originální Chou-Reggia smyčky je to 123 kompletních smyček, a to znamená nárůst o více jak 58%. Celkové urychlení tedy není tak špatné, a to především v důsledku použití počáteční vícenásobné sebe-replikace, díky které je navíc struktura růstu kolonie smyček pravidelného tvaru.



Obrázek 5.21: Kolonie urychlených smyček po 150 krocích.

6 Závěr

Hlavním cílem této práce je představení dvou různých přístupu k urychlení replikace sebe-replikujících se smyček. V rámci tohoto úkolu jsem provedl studii celulárních automatů. Dále jsem nastínil problematiku sebe-replikace struktur v celulárních automatech se zaměřením na struktury tvaru smyčky a jednotlivé smyčky stručně představil.

První jsem zvolil přístup návrhu pravidel celulárního automatu pomocí evolučních technik a pro tyto účely implementoval klasický genetický algoritmus. Tuto variantu jsem ale po rozvaze zavrhl a pravidla vedoucí k urychlení sebe-replikace navrhl experimentálním (ručním) přístupem. Zvolené dva kandidáty (Bylovu smyčku a Chou-Reggia smyčku) jsem urychlil pomocí dvou různých přístupů. První spočívající v obohacení smyček o schopnost vícenásobné sebe-replikace přinesl ve výsledku kromě mírného nárůstu počtu smyček hlavně změnu struktury růstu kolonie smyček. Druhý přístup je založený na rozšíření stavového prostoru, tj. rozšířil jsem počet stavů, které používají jednotlivé smyčky o 1, a tento nový stav použil k redukci počtu potřebných kroků nutných k vytvoření repliky smyčky. Tento přístup přinesl v případě Bylovy smyčky urychlení o 32% a v případě Chou-Reggia smyčky o cca 26,67%. Při kombinaci obou přístupů se mi podařilo po 150. kroku vývoje automatu ve srovnání s původními smyčkami dosáhnout v případě Bylovy smyčky více jak 73% nárůstu počtu replik smyček a v případě Chou-Reggia smyčky více jak 58% nárůstu počtu replik.

Přínosem prvního přístupu je především změna struktury růstu kolonie smyček. Počáteční sebe-replikace do všech směrů způsobuje, že vzor se stává pravidelným (čtvercový vzor), oproti původním smyčkám, kde je vzor spirálovitý. Přínosem druhého přístupu, především v kombinaci s prvním je výrazné urychlení nárůste počtu replik smyček. A opět díky počáteční sebe-replikaci do všech směrů nabývají kolonie smyček pravidelného vzoru.

Moderní sebe-replikující struktury bývají obdařeny schopností univerzálního výpočtu. Čím rychleji se dokážou replikovat, tím více výpočtů můžou provést. Urychlení jejich replikačních schopností má tedy určitě význam. Hodí se proto tuto oblast dále studovat a pokoušet se nalézt další přístupy vedoucí k stále lepším výsledkům. Například se pokusit ještě více rozšířit stavový prostor, buďto pomocí přidání dalšího stavu, anebo nahrazení menšího sousedství větším. Tyto přístupy mohou vést k obrovskému nárůstu stavového prostoru, což může vyústit v komplikace při hledání požadovaného řešení. A proto je dále potřeba se zamyslet nad možnými způsoby návrhu pravidel. Cesta k řešení může být v kombinaci více přístupů, např. evoluční návrh s experimentálním návrhem. Ale to jsou pouze teoretické úvahy, které je potřeba především ověřit praxí a jeví se být zajímavou cestičkou k prošlapání v oblasti sebe-replikace struktur.

Literatura

[1] Sipper, M.: *Evolution of Parallel Cellular Machines*. Lecture Notes in Computer Science, vol 1194, Springer-Verlag, Berlin Heidelberg New York, 1997. Dokument dostupný na URL http://www.moshesipper.com/pcm/ (prosinec 2009).

[2] Coveney, P., Highfield, R.: *Mezi chaosem a řádem*. Mladá fronta, Praha, 2003.

[3] Wolfram, S.: *A New Kind of Science*. Wolfram Media, 2002. Dokument dostupný na URL http://www.wolframscience.com/nksonline/toc.html (prosinec 2009).

[4] Weisstein, E. W.: *Moore Neighborhood*. From *MathWorld*--A Wolfram Web Resource. Dokument dostupný na URL http://mathworld.wolfram.com/MooreNeighborhood.html (prosinec 2009).

[5] Weisstein, E. W.: *von Neumann Neighborhood*. From *MathWorld*--A Wolfram Web Resource. Dokument dostupný na URL http://mathworld.wolfram.com/vonNeumann Neighborhood.html (prosinec 2009).

[6] *Cellular Automata/Neighborhood*. Wikibooks, the open-content textbooks collection. Dokument dostupný na URL http://en.wikibooks.org/wiki/Cellular_Automata/ Neighborhood (prosinec 2009).

[7] Bidlo, M.: *Biologií inspirovaný vývin jako technika evolučního návrhu*. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno.

[8] Peringer, P.: *SNT - Celulární automaty (doplněk)*. [Studijní materiál]. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno, 2009.

[9] Sekanina, L.: *Development v evolučním návrhu*. Biologií inspirované počítače. [Studijní materiál]. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno, 2009.

[10] Sipper, M.: *Fifty years of research on self-replication: An overview*. Artificial Life, 4 (3), 1998, s. 237–257.

[11] Sipper, M., Sanchez, E., Mange, D., Tomassini, M., P´erez-Uribe, A., Stauffer, A.: *Aphylogenetic, ontogenetic, and epigenetic view of bio-inspired hardware systems*. IEEE Transactions on Evolutionary Computation, 1 (1), 1997, s. 83–97.

[12] Chou, H.-H., Reggia, J. A.: *Emergence of self-replicating structures in a cellular automata space*. Physica D, 110 (3-4), 1997, s. 252-276.

[13] Drexler, K. E.: *Nanosystems: Molecular Machinery, Manufacturing and Computation*. John Wiley, New York, 1992.

[14] Freitas, R. A., Gilbreath, W. P.: *Advanced automation for space missions: Proceedings of the 1980NASA/ASEE summer study, Chapter 5: Replicating systems concepts: Self-replicating lunar factory and demonstrations*. NASA, Scientific and Technical Information Branch (available from U.S. G.P.O., Publication 2255), Washington D.C., 1980.

[15] Stauffer, A., Sipper, M.: *Emergence of Self-Replicating Loops in an Interactive, Hardware-Implemented Game-of-Life Environment*. Lecture Notes In Computer Science, Vol. 2493, Springer-Verlag, London, 2002, s. 123-131.

[16] Reggia, J. A., Lohn, J. D., Chou, H.-H.: *Self-replicating structures: evolution, emergence, and computation*. Artificial Life, 4 (3), 1998, s. 283–302.

[17] Sipper, M.: *The Artificial Self-Replication Page*. Dokument dostupný na URL http://www.cs.bgu.ac.il/~sipper/selfrep/ (prosinec 2009).

[18] Wolfram, S.: *Universality and Complexity in Cellular Automata*. Physica D, 10, 1984, s. 1-35.

[19] Sayama, H.: *Constructing evolutionary systems on a simple deterministic cellular automata space*. [Ph.D. disertation]. University of Tokyo, Department of Information Science, Graduate School of Science, 1998.

[20] Langton, Ch. G.: *Self-reproduction in cellular automata*. Physica D, 10 (1-2), 1984, s. 135-144.

[21] Reggia, J. A., Chou, H.-H., Armentrout, S. L., Peng Y.: *Minimizing Complexity in Cellular Automata Models of Self-Replication*. Proceedings of the 1st International Conference on Intelligent Systems for Molecular Biology, 1993, s. 337-344.

[22] Murray, R. K., Granner, D. K., Mayes, P. A., Rodwell, V. W.: *Harperova biochemie*. H & H, 4. vydání, 2002.

[23] Byl, J.: *On Cellular Automata and the Origin of Life*. Perspectives on Science and Christian Faith, 41, 1989, s. 26-29.

[24] Byl, J.: *Self-Reproduction in small cellular automata.* Physica D, 34, 1989, s. 295-299.

[25] Reggia, J. A., Armentrout, S. L., Chou, H.-H., Peng, Y.: *Simple systems that exhibit self-directed replication.* Science, 259, 1993, s. 1282-1287.

[26] Tempesti, G.: *A Self-Repairing Multiplexer-Based FPGA Inspired by Biological Processes.* [Ph.D. thesis]. Computer Science Department, Swiss Federal Institute of Technology Lausanne, 1998.

[27] Tempesti, G.: *A New Self-Reproducing Cellular Automaton Capable of Construction and Computation*. Lecture Notes in Computer Science, 929, 1995, s. 555–563.

[28] Perrier, J.-Y., Sipper, M., Zahnd, J.: *Toward a viable, self-reproducing universal computer*. Physica D, 97, 1996, s. 335–352.

[29] Wang, H.: *A variant to Turing's theory of computing machines*. Journal of the ACM, 4 (1), 1957, s. 63–92.

[30] Sayama, H.: *Introduction of Structural Dissolution into Langton's Self-Reproducing Loop*. Artificial Life VI: Proceedings of the Sixth International Conference on Artificial Life, Los Angeles, California. MIT Press, p. 114–122.

[31] Sayama, H.: *Spontaneous Evolution of Self-Reproducing Loops Implemented on Cellular Automata*: *A preliminary report*. Dept. of Information Science, Graduate School of Science, University of Tokyo, 1998.

[32] Oros, N., Nehaniv, C. L.: *Sexyloop: Self-Reproduction, Evolution and Sex in Cellular Automata*. The First IEEE Symposium on Artificial Life, Hawaii, USA, 2007, p.130-138.

[33] Oros, N., Nehaniv, C. L.: *Dude, Where is My Sex Gene? - Persistence of Sex over Evolutionary Time in Cellular Automata*. Nashville, TN, USA, 2009.

[34] Margulis, L, Sagan, D.: *Microcosmos: Four Billion Years of Evolution from Our Microbial Ancestors*. Harper Collins, 1987.

[35] Marsh, G. E.: *The Demystification of Emergent Behavior*. Argonne National Laboratory (Ret), 5433 East View Park, Chicago, IL 60615, 2009.

[36] Komenda, T.: *Sebereplikace v celulárních systémech*. [Diplomová práce]. Brno, FIT VUT v Brně, 2009.

[37] Schwarz, J., Sekanina. L.: *Aplikované evoluční algoritmy*. [Studijní opora]. Vysoké učení technické v Brně, Fakulta informačních technologií, Božetěchova 2, 612 66 Brno, 2006.

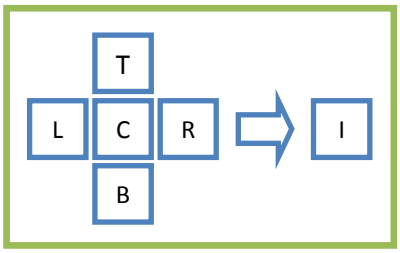
[38] Hynek, J.: *Genetické algoritmy a genetické programování*. Grada, Praha, 2008.

[39] Sayama, H.: *Self-Replication loops in Cellular Space*. Dokument dostupný na URL http://necsi.org/postdocs/sayama/sdsr/java/ (květen 2010).

Seznam příloh

Příloha 1. Originální pravidla Bylovy smyčky

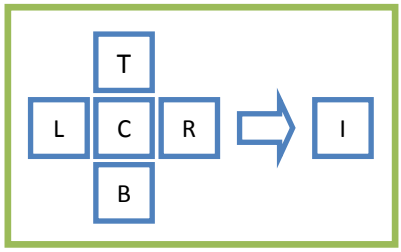
Počet pravidel je 238. Červeně jsou vyznačena pravidla, která byla při návrhu pravidel **upgrade č. 2 Bylovy smyčky** (viz. příloha 3) odebrána z původní sady pravidel.
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.



TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
30000->1	**10310->0**	**42101->4**	00251->5	34122->4	42103->4	**24504->2**
31000->5	20310->1	**00301->0**	02251->5	00222->0	**03103->0**	23414->3
12000->2	21310->1	**01301->0**	02351->2	**11322->1**	00303->0	00024->2
32000->3	23310->1	12301->1	12351->1	31322->1	12303->1	**10124->4**
42000->2	50020->5	32401->3	**23351->1**	13422->3	**00013->5**	30124->4
03000->1	20220->0	03501->2	01002->2	20522->0	42113->4	11124->4
13000->1	**52220->5**	42111->4	03002->3	02522->0	**12313->1**	**21124->4**
15000->2	15220->5	**00311->0**	04002->2	**03522->0**	22313->1	31124->4
25000->5	35220->3	**22311->1**	02202->0	42132->4	**52313->1**	23124->4
00100->0	15320->2	**23311->1**	**25202->5**	04132->4	23313->1	33124->4
10100->0	**52420->0**	32411->3	01302->1	14132->4	00513->2	00524->2
30100->3	40520->2	00021->2	03402->3	34132->4	00023->3	**32524->2**
01100->0	12520->4	00321->1	**25402->0**	50232->3	10123->3	24134->4
03100->3	22520->0	10321->1	21502->4	01332->1	44123->4	21334->1
33100->0	**44520->2**	30321->1	22502->0	**11332->1**	45123->3	32144->4
00200->0	00030->1	**21321->1**	**32502->0**	31332->1	14323->1	**02544->2**
22200->0	**10030->5**	**31321->1**	04502->2	**51332->5**	00423->3	32154->3
15200->5	20030->3	23321->1	**42112->4**	12332->1	10423->3	**21354->5**
10300->0	00130->3	33321->1	03112->3	**15332->1**	**50423->5**	01005->2
30300->0	**30130->0**	15321->1	**04112->4**	41432->3	11423->3	02005->5
01300->0	21130->3	**35321->5**	14112->4	**24532->2**	21423->3	01205->5
11300->0	00330->0	34421->3	50212->5	43142->4	51423->3	21205->5
12300->1	20430->3	02521->4	50312->2	53142->3	**20523->0**	23205->3
03300->0	21430->3	00031->1	01312->1	00542->2	**42523->2**	21305->2
32400->3	**25430->5**	21431->3	51312->1	**40542->2**	**00133->0**	**32405->5**
31500->2	10530->2	22431->3	**12312->1**	**23542->2**	42133->4	21315->1
42500->2	**22530->0**	52431->3	43312->1	00052->5	12333->1	**23315->5**
24500->2	20040->2	25431->3	**45312->5**	**20252->5**	22143->4	32415->3
20010->2	**21140->4**	21141->4	32412->3	**20452->0**	23143->4	**02225->5**
30010->1	23140->4	**22141->4**	03412->3	**03452->5**	12443->3	**31325->1**
50010->2	20540->2	23141->4	13412->3	13452->3	02253->3	**14325->5**
00110->0	**03001->5**	32341->1	35412->3	00003->1	**12353->5**	**02425->0**
50210->5	00101->0	**52341->5**	20512->4	01003->1	21453->3	13425->3
00310->0	32101->3	00051->2	**14122->4**	00103->3	02504->2	24135->3

Příloha 2. Pravidla upgrade č. 1 Bylovy smyčky

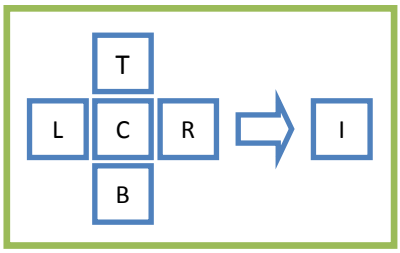
Počet pravidel je 252. Oproti originálním pravidlům Bylovy smyčky (viz. příloha 1) přibylo 14 pravidel (vyznačeny zeleně).
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.



TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
30000->1	21310->1	01301->0	12351->1	20522->0	22313->1	11124->4
31000->5	23310->1	12301->1	23351->1	02522->0	52313->1	21124->4
12000->2	50020->5	32401->3	01002->2	03522->0	23313->1	31124->4
32000->3	20220->0	03501->2	03002->3	42132->4	25313->1	23124->4
42000->2	52220->5	42111->4	04002->2	04132->4	00513->2	33124->4
03000->1	15220->5	00311->0	02202->0	14132->4	00023->3	53124->4
13000->1	35220->3	22311->1	25202->5	34132->4	10123->3	00524->2
15000->2	15320->2	23311->1	01302->1	50232->3	44123->4	32524->2
25000->5	52420->0	32411->3	03402->3	01332->1	45123->3	24134->4
00100->0	40520->2	00021->2	25402->0	11332->1	14323->1	21334->1
10100->0	12520->4	00321->1	21502->4	31332->1	00423->3	32144->4
30100->3	22520->0	10321->1	22502->0	51332->5	10423->3	02544->2
01100->0	44520->2	30321->1	32502->0	12332->1	50423->5	32154->3
03100->3	00030->1	21321->1	04502->2	15332->1	11423->3	21354->5
33100->0	10030->5	31321->1	42112->4	41432->3	21423->3	01005->2
00200->0	20030->3	23321->1	03112->3	24532->2	51423->3	02005->5
22200->0	00130->3	33321->1	04112->4	43142->4	20523->0	05005->5
15200->5	30130->0	15321->1	14112->4	53142->3	42523->2	01205->5
10300->0	21130->3	35321->5	50212->5	00542->2	00133->0	21205->5
30300->0	00330->0	34421->3	50312->2	40542->2	42133->4	23205->3
01300->0	20430->3	02521->4	01312->1	23542->2	12333->1	21305->2
11300->0	21430->3	00031->1	51312->1	00052->5	22143->4	32405->5
12300->1	25430->5	21431->3	12312->1	20252->5	23143->4	21315->1
03300->0	10530->2	22431->3	43312->1	20452->0	25143->4	23315->5
32400->3	22530->0	52431->3	45312->5	03452->5	12443->3	32415->3
31500->2	20040->2	25431->3	32412->3	13452->3	02253->3	35415->3
42500->2	21140->4	55431->3	03412->3	00003->1	05253->3	02225->5
24500->2	23140->4	21141->4	13412->3	01003->1	12353->5	31325->1
20010->2	20540->2	22141->4	35412->3	00103->3	21453->3	14325->5
30010->1	50050->5	23141->4	20512->4	42103->4	51453->3	02425->0
50010->2	35250->3	32341->1	14122->4	03103->0	02504->2	13425->3
00110->0	03001->5	52341->5	34122->4	00303->0	24504->2	42135->4
50210->5	00101->0	00051->2	00222->0	12303->1	23414->3	24135->3
00310->0	32101->3	00251->5	11322->1	00013->5	00024->2	50235->3
10310->0	42101->4	02251->5	31322->1	42113->4	10124->4	12335->1
20310->1	00301->0	02351->2	13422->3	12313->1	30124->4	00055->5

Příloha 3. Pravidla upgrade č. 2 Bylovy smyčky

Počet pravidel je 290. Oproti originálním pravidlům Bylovy smyčky (viz. příloha 1) přibylo 120 pravidel (vyznačeny zeleně), bylo pozměněno 9 pravidel (vyznačeny modře) a bylo odebráno 68 pravidel (červeně označená pravidla v příloze 1).
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.

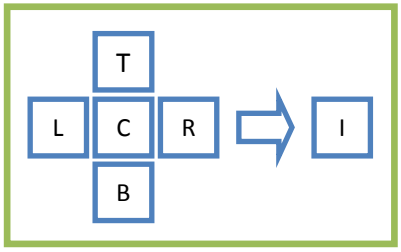


TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
30000->1	31220->5	03501->2	62302->0	41432->6	61423->3	01205->5
61000->6	22220->6	02601->2	03402->3	03632->5	30623->5	21205->5
12000->2	15220->5	42111->4	26402->0	43142->4	42133->4	23205->3
32000->3	35220->3	32411->3	21502->4	53142->3	31333->1	53205->3
42000->2	15320->2	00021->2	22502->0	63342->2	12333->1	21305->2
03000->1	62420->0	00321->1	04502->2	00542->2	13333->1	53305->2
13000->1	40520->2	10321->1	22602->5	00052->5	06043->4	21315->1
15000->2	12520->4	30321->1	03112->3	34152->4	22143->4	32415->3
25000->5	22520->0	23321->1	14112->4	31352->1	23143->4	35415->3
55000->5	10620->2	33321->1	50212->5	13452->3	25143->4	13425->3
00100->6	22620->5	53321->1	50312->2	34162->4	26143->4	42135->4
10100->0	00030->1	63321->1	01312->1	31362->1	66143->4	24135->3
30100->3	20030->3	15321->1	51312->1	20462->0	12443->6	50235->3
01100->0	00130->3	34421->6	43312->1	13462->3	02253->3	50335->2
22100->0	21130->3	02521->4	32412->3	00003->1	05253->3	12335->1
03100->3	00330->0	00031->1	03412->3	01003->1	05353->2	00055->5
00200->0	20430->3	33331->1	13412->3	00103->3	21453->3	13455->3
22200->1	21430->3	21431->3	35412->3	42103->4	51453->3	41006->6
15200->5	10530->2	22431->3	06412->3	00303->0	63004->4	01606->2
30300->0	23630->5	52431->3	20512->4	12303->1	02504->2	06606->0
12300->1	20040->2	25431->3	00612->2	66303->1	06604->2	00016->6
03300->0	16040->6	55431->3	00122->0	61403->3	60014->6	30416->3
32400->3	23140->4	06431->3	34122->4	32603->5	66014->3	32416->3
31500->2	20540->2	26431->3	00222->1	42113->4	23414->6	20326->0
42500->2	50050->5	21141->4	20222->6	02213->5	00024->2	10426->3
24500->2	35250->3	23141->4	02222->6	22313->1	30124->4	02426->0
21600->2	35350->2	32341->1	31322->1	23313->1	11124->4	40036->4
64600->2	10060->6	00051->2	06322->0	33313->1	31124->4	42136->4
16600->2	34060->4	00251->5	13422->3	25313->1	23124->4	46136->4
66600->0	22360->0	02251->5	20522->0	26313->1	33124->4	12336->1
20010->2	63360->1	02351->2	02522->0	00513->2	53124->4	24336->2
30010->1	21460->3	12351->1	20622->5	00023->3	63124->4	06336->1
50010->2	13460->3	04061->6	02622->5	10123->3	00524->2	16046->3
00110->0	40660->2	64061->3	42132->4	44123->4	24134->4	00646->2
50210->5	60660->0	00661->2	04132->4	45123->3	21334->1	41066->3
20310->1	06001->6	01002->2	14132->4	14323->1	32144->4	34166->4
21310->1	00101->0	03002->3	34132->4	46323->2	32154->3	30366->1

23310->1	32101->3	04002->2	**10232->5**	00423->3	**63164->4**	**00666->0**
60610->2	**23201->5**	**02102->0**	50232->3	10423->3	**32364->2**	
50020->5	12301->1	**02202->1**	01332->1	11423->3	01005->2	
20120->0	32401->3	**22202->6**	31332->1	21423->3	02005->5	
20220->1	**62401->3**	01302->1	12332->1	51423->3	**05005->5**	

Příloha 4. Originální pravidla Chou-Reggia smyčky

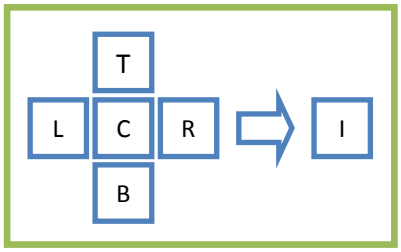
Počet pravidel je 137. Červeně jsou vyznačena pravidla, která byla při návrhu pravidel **upgrade č. 2 Chou-Reggia smyčky** (viz. příloha 6) odebrána z původní sady pravidel.
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.



TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
50000->2	40110->4	**33530->0**	04301->1	00141->4	12322->1	00314->1
12000->4	04110->4	**50040->2**	44301->1	**40141->4**	10403->3	**20314->1**
22000->2	14110->4	**44040->5**	**54301->1**	01141->4	01403->3	**50314->1**
05000->2	**44110->4**	00140->5	**45301->0**	**04141->4**	11403->3	04314->1
45000->2	**20210->0**	01140->4	**00401->1**	**04351->1**	**13403->3**	**05314->1**
40100->5	**12210->0**	**14140->4**	30401->3	**01002->4**	00503->3	**40044->5**
04100->5	04310->1	**12240->1**	40111->4	02002->2	**33503->0**	**04044->5**
14100->4	05310->1	10340->1	**20211->0**	**01202->0**	**01433->3**	**01144->4**
00200->0	**00410->1**	01340->1	30411->3	**11202->0**	**30533->0**	10344->1
21200->0	**20410->3**	41340->1	**00021->4**	**21202->1**	**03533->0**	**00054->2**
12200->0	30410->3	**51340->0**	**00221->0**	**41202->1**	**44004->5**	**10354->0**
22200->0	03410->3	**12340->1**	**01221->0**	**02202->0**	00104->5	00005->2
41300->1	13410->3	**15340->1**	**02221->1**	**41302->1**	10104->4	**04005->2**
10400->1	**33410->3**	00050->2	**04221->1**	**01402->3**	01104->4	10305->1
01400->1	20020->2	01350->1	22321->1	**00212->0**	11104->4	**41305->1**
12400->3	**10220->0**	**41350->1**	**04321->1**	**20212->1**	**41104->4**	**04315->0**
13400->3	**20220->0**	40101->4	**00421->3**	22312->1	**14104->4**	**10345->1**
30500->3	**12220->1**	**02201->0**	00431->3	00022->2	10304->1	
03500->3	01430->3	40301->1	**30431->3**	**00222->0**	**40114->4**	
20010->4	00530->3	50301->1	01431->3	21322->1	**20214->1**	

Příloha 5. Pravidla upgrade č. 1
Chou-Reggia smyčky

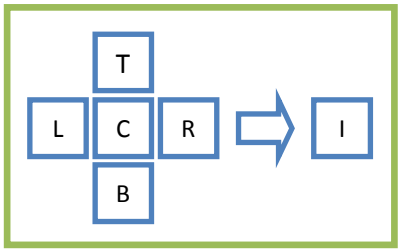
Počet pravidel je 144. Oproti originálním pravidlům Chou-Reggia smyčky (viz. příloha 4) přibylo 7 pravidel (vyznačeny zeleně).
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.



TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
50000->2	04110->4	44040->5	45301->0	04141->4	01403->3	04314->1
12000->4	14110->4	00140->5	00401->1	10341->1	11403->3	54314->1
22000->2	44110->4	01140->4	30401->3	04351->1	13403->3	05314->1
05000->2	20210->0	14140->4	40111->4	44351->1	00503->3	40044->5
45000->2	12210->0	12240->1	20211->0	01002->4	33503->0	04044->5
40100->5	04310->1	10340->1	30411->3	02002->2	01433->3	01144->4
04100->5	05310->1	01340->1	00021->4	01202->0	30533->0	10344->1
14100->4	00410->1	41340->1	00221->0	11202->0	03533->0	15344->1
00200->0	20410->3	51340->0	01221->0	21202->1	44004->5	00054->2
21200->0	30410->3	12340->1	02221->1	41202->1	00104->5	10354->0
12200->0	03410->3	15340->1	04221->1	02202->0	10104->4	12354->0
22200->0	13410->3	00050->2	22321->1	41302->1	01104->4	00005->2
41300->1	33410->3	01350->1	04321->1	01402->3	11104->4	04005->2
10400->1	20020->2	41350->1	45321->0	00212->0	41104->4	10305->1
01400->1	10220->0	40101->4	00421->3	20212->1	14104->4	41305->1
12400->3	20220->0	02201->0	00431->3	22312->1	10304->1	04315->0
13400->3	12220->1	40301->1	30431->3	00022->2	40114->4	24315->0
30500->3	01430->3	50301->1	01431->3	00222->0	20214->1	10345->1
03500->3	00530->3	04301->1	00141->4	21322->1	00314->1	
20010->4	33530->0	44301->1	40141->4	12322->1	20314->1	
40110->4	50040->2	54301->1	01141->4	10403->3	50314->1	

Příloha 6. Pravidla upgrade č. 2
Chou-Reggia smyčky

Počet pravidel je 217. Oproti originálním pravidlům Chou-Reggia smyčky (viz. příloha 4) přibylo 156 pravidel (vyznačeny zeleně), bylo pozměněno 8 pravidel (vyznačeny modře) a bylo odebráno 76 pravidel (červeně označená pravidla v příloze 4).
Pravidla jsou uvedena podle schématu zobrazeného na následujícím obrázku.



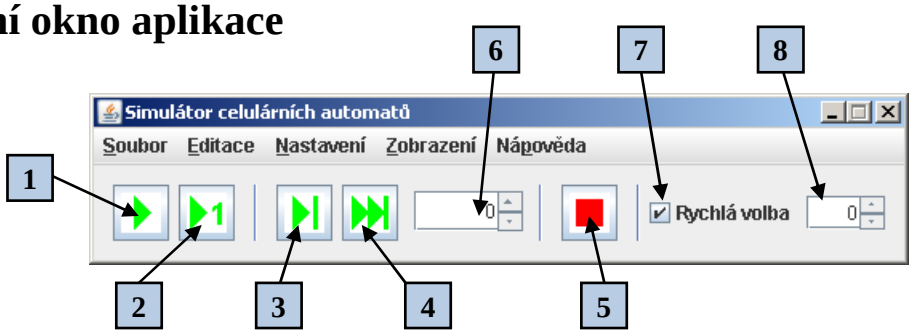
TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I	TLCRB->I
50000->2	41310->1	60640->4	10601->1	22112->5	61104->4	40306->1
22000->6	04310->6	01640->0	40601->0	22312->1	10304->6	41306->1
32000->5	05310->1	00050->2	06601->0	00022->6	60304->1	30406->3
62000->6	30410->3	14150->4	40111->4	21122->5	40504->0	03406->1
05000->2	03410->3	00250->6	30411->3	12122->5	50504->4	13406->3
26000->6	13410->3	01350->1	60611->0	21322->1	10604->0	46506->4
36000->1	35410->3	03450->3	22121->5	12322->1	00314->1	11606->0
40100->5	36410->3	13450->3	22321->1	00062->6	60314->1	61606->1
04100->5	00610->0	04550->4	00431->3	16562->1	01314->1	04606->4
14100->4	01610->1	20060->6	50431->3	46562->4	04314->1	64606->5
00200->0	04610->0	14160->4	60431->3	10403->3	65314->1	01016->5
50200->6	16610->0	10260->0	01431->3	50403->3	10344->1	40116->4
61200->0	20020->6	16260->5	05431->3	60403->3	60564->4	00216->0
05200->6	60020->6	04360->1	06431->3	01403->3	26564->4	60216->5
16200->0	20030->5	03460->3	56431->3	11403->3	00664->4	30416->3
41300->1	60030->1	13460->3	00141->4	51403->3	06664->5	35416->3
13400->3	60430->1	10660->0	01141->4	61403->3	00005->2	62516->1
36400->1	01430->3	30660->0	05141->4	00503->3	00205->6	00616->0
30500->3	05430->3	50660->0	06141->4	06603->0	10305->1	60616->1
03500->3	06430->3	16660->1	10341->1	00023->5	30405->3	00026->6
10600->0	00530->3	46660->5	10061->5	01453->3	13405->3	61526->1
01600->0	00140->5	16001->5	00261->0	61453->3	40505->4	64526->4
61600->0	01140->4	40101->4	06261->5	00063->1	06605->0	66436->3
63600->0	10340->1	06201->0	04361->1	00463->1	40115->4	00636->0
65600->0	01340->6	40301->6	54361->1	01463->3	30415->3	62546->4
46600->4	41340->1	50301->1	26561->1	66463->3	16345->1	60646->5
61010->5	06340->1	04301->1	01661->0	00104->5	13465->3	41356->1
40110->4	16340->1	14301->1	06661->1	10104->4	02006->6	00656->0
04110->4	04540->0	44301->1	02002->6	01104->4	03006->1	63466->3
14110->4	05540->4	30401->3	03002->5	11104->4	01206->0	36466->3
60210->0	66540->4	00601->0	06002->6	51104->4	61206->5	04566->4

Příloha 7. Popis a návod k použití programu CASimulator2D

Tento simulátor je implementován v jazyce *Java* ve verzi *Java SE 1.6.0 update 20*. Pro provoz aplikace je nutné mít nainstalované alespoň *Java SE Runtime Environment 6*. Program lze stáhnout ze stránek http://code.google.com/p/ca-simulator-2d/. Spouští se z příkazové řádky pomocí příkazu:

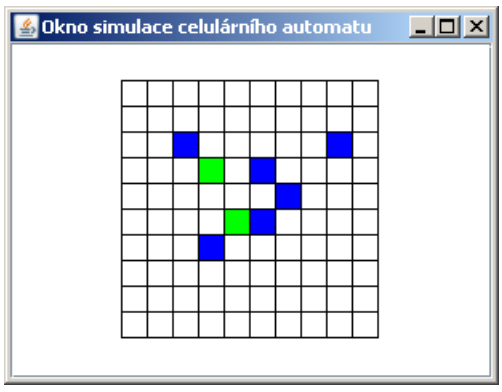
java -jar CASimulator2D.jar.

Hlavní okno aplikace

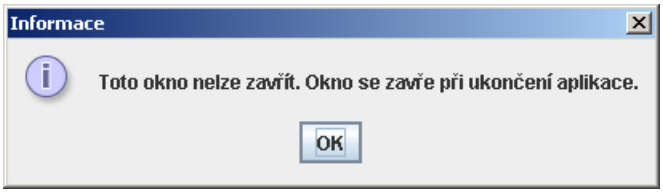


1. Spouští/přerušuje simulaci. Rychlost simulace lze nastavit pomocí Nastavení -> Rychlost simulace.
2. Provede jeden krok simulace.
3. Spustí simulaci na počet kroků zadaných v textovém poli (6). Lze přerušit zmáčknutím (1).
4. Skočí v simulaci o počet kroků zadaných v textovém poli (6).
5. Přerušuje simulaci a zároveň vrací stav buněk v simulaci do nějakého uloženého počátečního stavu.
6. Textové pole určující počet kroků, které se mají odsimulovat.
7. Aktivuje možnost upravovat stavy buněk v *oknu simulace*.
8. Výběr stavu, který se má při kliknutí levým tlačítkem myši na buňku v *okně simulace* nastavit.

Okno simulace

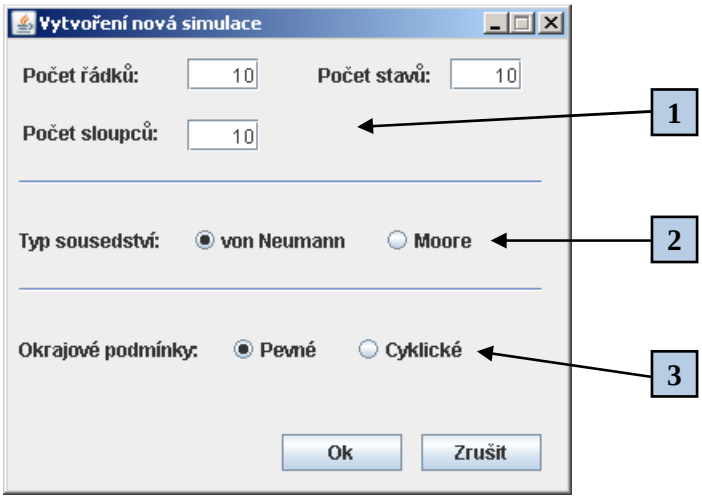


Okno, ve kterém probíhá vizualizace průběhu simulace CA. Pokud je zvolena v *hlavním okně* aplikace možnost Rychlá volba (7), pak po kliknutím levým tlačítkem myši na buňku se nastaví hodnota stavu buňky na hodnotu zadanou v poli (8). Po kliknutí pravým tlačítkem myši se změní stav buňky na: „aktuální stav“ + 1 **modulo** „počet stavů“. Okno nelze zavřít. Zavírá se s ukončením aplikace. Pokud se o to uživatel pokusí, vyhodí se následující dialog:



Menu Soubor

- *Nový* - Otevře nové okno pro vytvoření nové simulace.
    1. Výběr počtu řádků, sloupců mřížky automatu a počtu stavů na buňku.
    2. Výběr ze dvou typů sousedství. Obě sousedství jsou napevno pro rádius 1.
    3. Výběr ze dvou typů okrajových podmínek. U pevného je napevno nastavena hodnota stavu mimo mřížku na hodnotu 0.

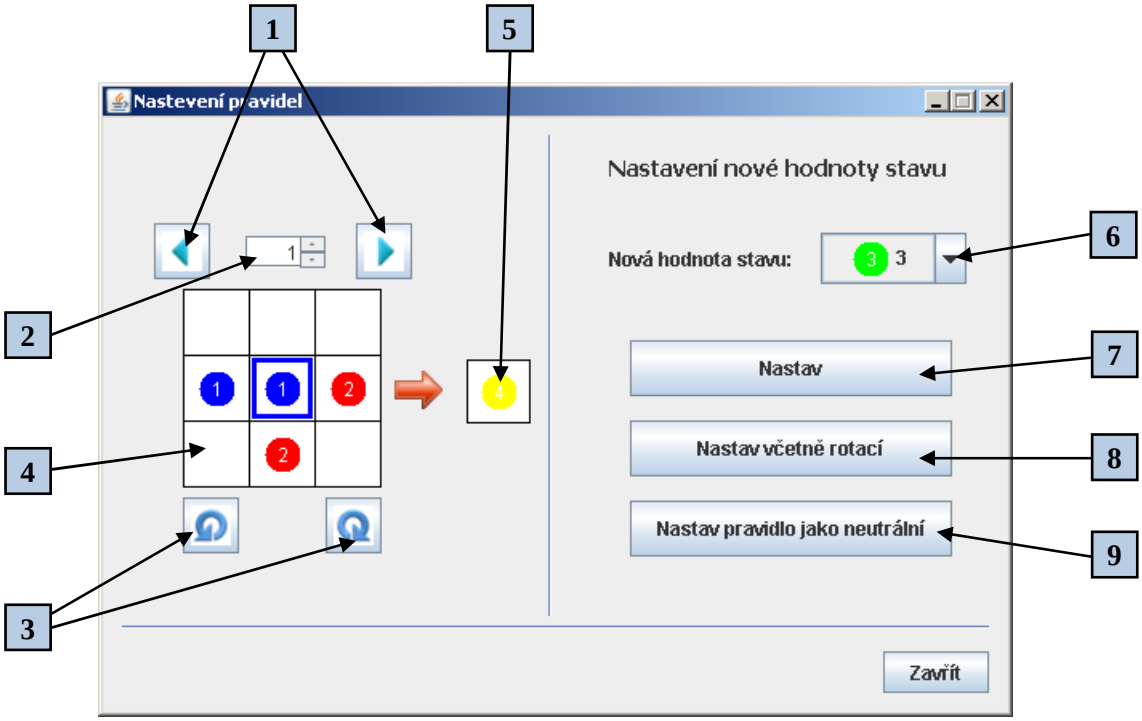


- *Otevřít* - Nahraje ze souboru uloženou konfiguraci a pravidla CA.
- *Uložit jako* - Uloží do souboru konfiguraci a pravidla CA.
- *Uložit jako, redukovaně* - Uloží do souboru konfiguraci a pravidla CA v redukované podobě, tj. pouze aktivní pravidla.
- *Konec* - Ukončení aplikace.

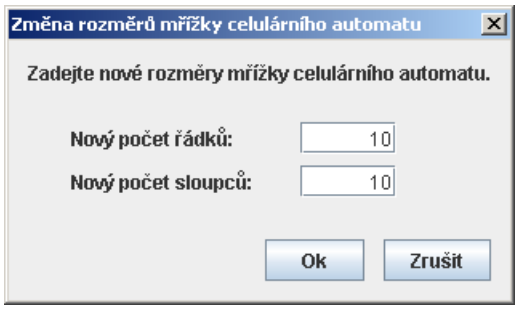
Menu Editace

- *Edituj pravidla* - Otevře okno pro editaci pravidel.
    1. Tlačítka pro přepnutí na předchozí, resp. následující pravidlo.
    2. Textové pole ukazující hodnotu stavu vybrané buňky (viz. (4)) a pro nastavení stavu vybrané buňky.

3. Tlačítka pro rotaci pravidla proti, resp. po směru hodinových ručiček.
4. Plocha pro zobrazování a nastavování kombinace okolí. Levým tlačítkem myši se vybírá buňky (orámuje se modře) a poté lze měnit hodnotu stavu pomocí (2). Pravým tlačítkem myši se mění hodnota stavu přímo podle schématu „aktuální stav“ + 1 **modulo** „počet stavů“.
5. Hodnota pravidla pro zvolené okolí v (4). Levým tlačítkem myši se hodnota stavu dekrementuje, pravým tlačítkem se inkrementuje.
6. Combo box pro určení hodnoty stavu.
7. Nastaví pro zvolené okolí (4) hodnotu v (5) na stav zadaný v (6).
8. Nastaví pro zvolené okolí (4) hodnotu v (5) na stav zadaný v (6) včetně všech možných rotací.
9. Nastaví pro zvolené okolí (4) hodnotu v (5) na stav středové buňky v (4).



- *Změnit velikost mřížky* - Otevře dialogové okno pro změnu velikosti mřížky (počet řádků a sloupců). Pokud se rozměr zvětšuje, tak nově přidané buňky mají stav 0.

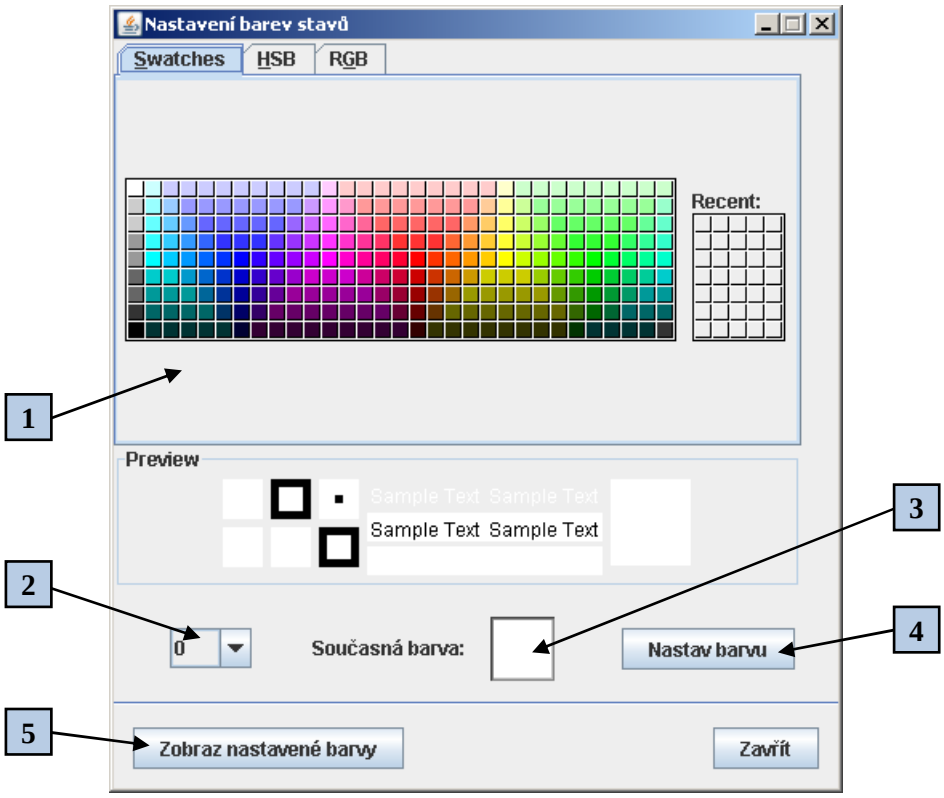


- *Zapamatovat aktuální stavy* - Uloží aktuální stavy buněk. Tyto stavy se nahrají kliknutím v hlavním okně aplikace na tlačítko (5).

Menu Nastavení

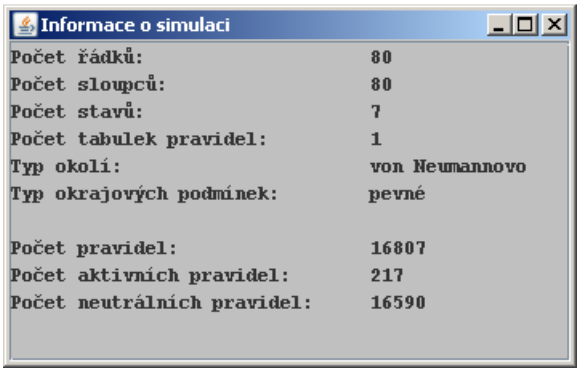
- *Rychlost simulace* - Otevře okno pro nastavování rychlosti průběhu simulace.

- *Nastavení barev stavů* – Otevře okno pro nastavování barev stavů.
    1. Paleta barev. Zde se volí barva, která se nastaví vybranému stavu v (2) stisknutím tlačítka (4).
    2. Vybraný stav.
    3. Aktuálně nastavená barva zvoleného stavu v (2).
    4. Nastaví barvu stavu v (2) na novou barvu vybranou v (1).
    5. Otevře jednoduché okno, které zobrazuje nastavené barvy pro všechny stavy z aktuální simulace.



Menu Zobrazení

V prostém okně zobrazuje informace o aktuální simulaci.



Příloha 8. Demonstrace výsledků práce

Tato příloha má za úkol vysvětlit, jak demonstrovat výsledky uložené na přiloženém médiu CD-R. Výsledky se otevírají a simulují pomocí programu CASimulator2D, který se nachází v adresáři *simulator_CA*. Stejný adresář obsahuje i uživatelskou příručku k programu, která je zároveň součástí této práce jako příloha 7. Samotné výsledky jsou uloženy v adresáři *vysledky_prace*.

Adresář *vysledky_prace*/*urychleni_sebe-replikace* obsahuje v podadresáři pro_CASimulator2D 4 soubory s nově nalezenými pravidly a 2 soubory s původními pravidly smyček spustitelné v programu CASimulator2D. Význam jednotlivých souborů shrnuje následující tabulka.

Soubor	Obsah
byl_original	Originální pravidla Bylovy smyčky
byl_upgrade_1	Pravidla urychlené smyčky, viz. kap. 5.1
byl_upgrade_2	Pravidla urychlené smyčky, viz. kap. 5.2
chou_reggia_original	Originální pravidla Chou-Reggia smyčky
chou_reggia_upgrade_1	Pravidla urychlené smyčky, viz. kap. 5.3
chou_reggia_upgrade_2	Pravidla urychlené smyčky, viz. kap. 5.4

Všechny soubory mají shodně mřížku buněk o velikost 80x80 buněk. Sebe-replikující se struktury se nacházejí přibližně na středu této mřížky. Pro lepší přehled lze změnit velikost mřížky buněk pomocí menu *Zobrazení -> Velikost CA*.

Adresář *vysledky_prace*/*evolucni_navrh_pravidel* obsahuje v 2 soubory s pomocí evoluce nalezenými pravidly (viz. kap. 4.3) spustitelné v programu CASimulator2D. Význam jednotlivých souborů shrnuje následující tabulka.

Soubor	Obsah
cross_rules	Pravidla replikace jednoduché struktury tvaru kříže
byl_rules004	Pravidla simulující prvních 21 kroků vývoje Bylovy smyčky

V případě prvního souboru je velikost mřížky 100x100 buněk a struktura se opět nachází přibližně na středu mřížky. Druhý soubor má velikost mřížky pouze nezbytně nutnou k demonstraci funkčnosti prvních 21 kroků vývoje Bylovy smyčky.